

4-2016

A study of how Chinese ink painting features can be applied to 3D scenes and models in real-time rendering

Muning Cao
Purdue University

Follow this and additional works at: https://docs.lib.purdue.edu/open_access_theses



Part of the [Computer Sciences Commons](#)

Recommended Citation

Cao, Muning, "A study of how Chinese ink painting features can be applied to 3D scenes and models in real-time rendering" (2016).
Open Access Theses. 753.
https://docs.lib.purdue.edu/open_access_theses/753

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

**PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance**

This is to certify that the thesis/dissertation prepared

By Muning Cao

Entitled

A STUDY OF HOW CHINESE INK PAINTING FEATURES CAN BE APPLIED TO 3D SCENES AND MODELS IN
REAL-TIME RENDERING

For the degree of Master of Science

Is approved by the final examining committee:

Tim McGraw

Chair

David Whittinghill

Co-chair

Rick Paul

Co-chair

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Tim McGraw

Approved by: Regina M. Brown

Head of the Departmental Graduate Program

4/20/2016

Date

A STUDY OF HOW CHINESE INK PAINTING FEATURES CAN BE APPLIED TO
3D SCENES AND MODELS IN REAL-TIME RENDERING

A Thesis

Submitted to the Faculty

of

Purdue University

by

Muning Cao

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

May 2016

Purdue University

West Lafayette, Indiana

TABLE OF CONTENTS

	Page
LIST OF TABLES	v
LIST OF FIGURES	vi
ABSTRACT	viii
CHAPTER 1. INTRODUCTION	1
1.1 Background	1
1.2 Statement of Purpose	2
1.3 Significance	3
1.4 Definitions	4
1.5 Assumptions	5
1.6 Limitations	5
1.7 Delimitations	6
1.8 Summary	7
CHAPTER 2. REVIEW OF LITERATURE	8
2.1 Chinese Ink Painting (2D / 3D)	8
2.2 Non-photorealistic Rendering (NPR)	11
2.3 Image-based Rendering	14
2.4 Virtual Painting	16
2.5 Silhouette Detection	19
2.6 Pipeline of Geometry-based NPR	21
2.7 Neural Network (NN)	22
2.8 Summary	24
CHAPTER 3. METHODOLOGY	25
3.1 Development Tools	25

	Page
3.2 Framework	25
3.3 Rendering Pipeline	28
3.3.1 Pipeline for rendering static objects	28
3.3.2 Pipeline for rendering moving objects.....	29
3.3.3 Interior Shading	31
3.3.4 Silhouette Shading	32
3.3.5 Static Texture.....	33
3.3.6 Skeleton Animation	34
3.4 Assessment Instruments	35
3.5 Summary	35
CHAPTER 4. IMPLEMENTATION.....	36
4.1 Interior and Silhouette Shading.....	36
4.1.1 Basic Interior and Silhouette Shading	36
4.1.2 Gaussian Blur.....	37
4.1.3 Smooth Step Function.....	38
4.1.4 Ink Texture.....	39
4.2 Other Experiments.....	40
4.2.1 Stroke Structure	41
4.2.2 Watercolor Style	42
4.3 Summary	43
CHAPTER 5. RESULTS	44
5.1 Rendering	44
5.1.1 Single Meshes.....	44
5.1.2 Final Scene.....	45
5.2 Testing.....	46
5.3 Summary	48
CHAPTER 6. CONCLUSIONS.....	49
6.1 Summary of the Research	49
6.2 Future Work	49

	Page
LIST OF REFERENCES	51
APPENDICES	
Appendix A Figure 4.6 with high resolution	54
Appendix B Results of Rendering Single Meshes	55
Appendix C Gaussian Blur.....	61
Appendix D Multi-pass Rendering	63
Appendix E Mesh Information.....	64

LIST OF TABLES

Table	Page
Table 1 Speed testing	46
Table 2 Rendering single meshes	55

LIST OF FIGURES

Figure	Page
Figure 2.1 Examples of a stroke and line	9
Figure 2.2 Mountains painted in different styles	10
Figure 2.3 Example of ink bleeding	10
Figure 2.4 Comparison of real and NPR artworks (Isenberg et al., 2006)	12
Figure 2.5 Different NPR features (Cartoon and sketch)	13
Figure 2.6 Different NPR features (Oil painting).....	13
Figure 2.7 An example of real-time Chinese ink painting rendering algorithm (Dong et al., 2014).....	16
Figure 2.8 Anatomy of a typical brush (Nelson S H Chu & Tai, 2004)	17
Figure 2.9 Example of the process of implementation of a real-time NPR drawing system (Kalnins et al., 2002).....	18
Figure 2.10 Example of the details of a real-time NPR drawing system (Chen et al., 2015)	18
Figure 2.11 Basic silhouette detection methods (Hertzmann, 1999)	19
Figure 2.12 Sample of rendering process: the three passes of Yuan's whole rendering process (Yuan et al., 2007).....	21
Figure 2.13 A general structure of an artificial neural network.....	23
Figure 3.1 Elements to be rendered in a scene	26
Figure 3.2 Design of the proposed project	27
Figure 3.3 Testing of the proposed project.....	28
Figure 3.4 Pipeline for rendering static objects	28
Figure 3.5 Pipeline for rendering moving objects	29
Figure 3.6 1D Gaussian blur.....	31

Figure	Page
Figure 3.7 Noise textures.....	33
Figure 3.8 The GLSL function smoothstep(Edge0, Edge1, x)	34
Figure 4.1 Performance of basic interior rendering and silhouette rendering	36
Figure 4.2 Comparison of features with different “threshold” value.....	37
Figure 4.3 Silhouette Gaussian blur with different kernel width.....	38
Figure 4.4 Effect of varying smoothstep edge parameters	39
Figure 4.5 Silhouette with different noise textures attached	40
Figure 4.6 Stroke structure implementation	41
Figure 4.7 Watercolor style rendering.....	42
Figure 5.1 Original models.....	44
Figure 5.2 Models rendered with the pipeline proposed by this project.....	45
Figure 5.3 Final static scene	45
Figure 5.4 Speed testing with different number of vertices.....	46
Figure 5.5 Speeding testing on screens with different resolution.....	47

ABSTRACT

Cao, Muning. M.S., Purdue University, May 2016. A Study Of How Chinese Ink Painting Features Can Be Applied To 3D Scenes And Models In Real-time Rendering. Major Professor: Tim McGraw.

Past research findings addressed mature techniques for non-photorealistic rendering. However, research findings indicate that there is little information dealing with efficient methods to simulate Chinese ink painting features in rendering 3D scenes. Considering that Chinese ink painting has achieved many worldwide awards, the potential to effectively and automatically develop 3D animations and games in this style indicates a need for the development of appropriate technology for the future market.

The goal of this research is about rendering 3D meshes in a Chinese ink painting style which is both appealing and realistic. Specifically, how can the output image appear similar to a hand-drawn Chinese ink painting. And how efficient does the rendering pipeline have to be to result in a real-time scene.

For this study the researcher designed two rendering pipelines for static objects and moving objects in the final scene. The entire rendering process includes interior shading, silhouette extracting, textures integrating, and background rendering. Methodology involved the use of silhouette detection, multiple rendering passes, Gaussian blur for anti-aliasing, smooth step functions, and noise

textures for simulating ink textures. Based on the output of each rendering pipeline, rendering process of the scene with best looking of Chinese ink painting style is illustrated in detail.

The speed of the rendering pipeline proposed by this research was tested. The framerate of the final scenes created with this pipeline was higher than 30fps, a level considered to be real-time. One can conclude that the main objective of the research study was met even though other methods for generating Chinese ink painting rendering are available and should be explored.

CHAPTER 1. INTRODUCTION

The goal of this research is about rendering 3D meshes in a Chinese ink painting style which is both appealing and realistic. Past research findings addressed mature techniques for NPR. Some of these research projects presented geometry-based rendering in art styles similar to ink painting styles including good NPR lighting models to be applied to different art styles. However, there was no study that integrates everything together to analyze a broad range of scenes rendered in a Chinese ink painting style. This research project focuses mainly on the gap between the existing and the integrated methodology used.

This chapter establishes the basis of the research study and includes a brief description of the background, purpose, and significance of the research effort. The chapter also provides the assumptions, limitations and delimitations of the study.

1.1 Background

Chinese ink painting is one of the oldest continuous artistic traditions in the world. Through the development of Chinese ink painting, refining the brush movement and ink flow is at the core of this type of art. In the ink painting style, ink density varies to affect tonality and shading. In the hand of an ink painting artist, the performance changes both by differential grinding of the ink stick in water and by varying the ink load and pressure within a single brushstroke.

To apply Chinese ink painting to animations and games with changing motion and lighting, a technique called non-photorealistic rendering (NPR) is needed. NPR is an area of computer graphics for achieving a wide variety of expressive styles in digital art. It can use both 2D image-based and 3D geometry-based approaches. An image-based technique is the type to be applied in this research. It is most commonly seen in video games and movies. This technique involves modification of a 3D model from its original input to a new artistic style where in the geometry of the model is kept, but some content and details are portrayed in a different way. The result is that after applying NPR techniques, the scene with all 3D objects will appear in a 2D format.

1.2 Statement of Purpose

The purpose of this research is to present a project involving rendering 3D meshes in a Chinese ink painting style. It is a significant challenge to apply Chinese ink painting style to 3D models. In previous studies there were algorithms of real-time painting with different brushes and rendering strokes directly on 3D models. But the researcher is concerned about the real-time lighting and motions of 3D objects rendered with Chinese ink painting features.

Chinese ink painting is characterized not only by the shape and color of the object, but also by brush strokes and color bleeding. Although not photorealistic, it is different from most rendering techniques that only pay attention to points and edges. In this study, the researcher will develop an approach to render a scene featuring an ink painting style.

The real-time 3D rendering techniques to be applied in this research project can be developed as a GPU-based rendering by applying some existing libraries such as Open

Graphics Library (OpenGL). The outcome of this research will present a process of automatically converting 3D triangle meshes into appealing Chinese ink painting style representations. The entire rendering process may include silhouette extraction, the transformation from point-edge to strokes, and simulating ink bleeding in shaders. Moreover, an appropriate lighting model will be applied for such 2D features in 3D scenes. The creation of an ink-painting feature may be time-consuming, but optimizing the existing algorithms for rendering is important to improve the efficiency and performance of the result.

1.3 Significance

Historically, research in computer graphics primarily paid attention to producing images that are indistinguishable from photographs. However, not all visual content is appropriate to be presented by photographs. Graphic designers need more choices to portray visual information in a better way. Sometimes when images are required to delineate specific themes, photorealistic performance has to be simplified to a clean and non-photorealistic format.

Industry wide development of Chinese ink painting in industry of 2D rendering and animation is mature. A great number of creative designs and products have already been produced. However, the techniques used were too time-consuming to be applied in 3D scenes. Research findings indicate that there is little available information dealing with efficient methods to simulate brush strokes and the ink bleeding. Therefore, such features seldom occurred in 3D applications even though the 3D rendering industry has developed quickly in recent years. In contrast to the large-scale implementation of computer graphics in rendering western art styles, little research has been done involving

the technology to help artists and programmers to efficiently create 3D scenes rendered with Chinese ink painting features.

Consider that Chinese ink painting has achieved many worldwide awards. For example, many products in Chinese ink painting style were exhibited at the Fine Art Asia 2015 and the Milan Expo and awarded gold prizes. Thus the potential to effectively and automatically develop 3D animations and games in this style indicates a need for the development of appropriate technology for the future market.

1.4 Definitions

Vertex: A common vertex is a vertex adjacent to 3 or more edges. Its adjacent faces have to be triangles. A vertex in this research project needs to hold at least 3 attributes namely position attribute, normal attribute and texture coordinate attribute.

Cusp: “A vertex is called a cusp vertex (or cusp) if one of the following holds:

- (i) It is adjacent to exactly 2 silhouette edges, one front-facing and the other back-facing,
- (ii) It is adjacent to more than 2 silhouette edges, or 3 and is adjacent to a border edge.” (Dooley & Cohen, 1990)

Polygon: “A polygon is front-facing if the dot product of its outward normal and a vector from a point on the polygon to the camera position is positive. Otherwise the polygon is back-facing.” (Markosian et al., 1997)

Silhouette: A silhouette edge is an edge adjacent to one front-facing and one back-facing polygon. A border edge is an edge adjacent to just one face. (Markosian et al., 1997). A silhouette face is a face adjacent to a silhouette edge or adjacent to another

silhouette face. A silhouette is a field containing several silhouette faces which satisfy prescribed conditions.

View: “A view is generic if:

- (i) the multiplicity of the image of the silhouette curves is everywhere one, except at a finite number of points where it is two;
- (ii) these multiplicity-two points do not coincide with the projection of any vertices of the mesh;

their number is invariant under small changes in viewing direction.” (Appel, 1967)

Mesh: A 3D mesh consists of acceptable vertices and triangle faces. An acceptable mesh has no cusps and border edges.

1.5 Assumptions

The assumptions in this thesis include:

- The target art feature to be applied in this research involves Chinese ink painting.
- The devices for testing the project works repeatedly. The efficiency of rendering different meshes and features can be measured and compared directly by analyzing the rendering speed.

1.6 Limitations

The limitations in this thesis include:

- The research project will apply an existing approach of silhouette detection for 3D models.

- The research project will apply an existing lighting model for non-photorealistic rendering. It is based on making some modifications in the lighting model to make it fit the features better.
- The research project will develop a real-time rendering project mainly with OpenGL. The pipelines will be designed based on OpenGL.
- Only the efficiency and performance of rendering different styles of strokes will be tested in this research project.
- The research project will find the difference between the efficiency of rendering simple meshes and complex meshes (meshes with a great number of vertices).
- The research project will modify the algorithm trying to make the project more efficient.
- The research project will only use 3D meshes.

1.7 Delimitations

The delimitations in this thesis include:

- The research project will not develop new silhouette detection approaches.
- The research project will not develop a new lighting model.
- OpenGL already made great efforts on rendering 3D scenes. The research project will not explain the algorithm and logic done by OpenGL.
- Chinese ink painting style is very casual and personalized. The result of this project will not have the randomness that real ink art products have.
- The research project will not test the performance of different color schemes.

- The research project will not generate a general optimization for NPR rendering algorithms.
- The research project will not work correctly when the input 3D meshes have cusps or single bounding edges.

1.8 Summary

This chapter set the foundation for developing a real-time rendering project. Information provided in this chapter includes background, purpose of the research, significance, and boundaries assigned to the research project. The next chapter examines existing studies with different perspectives involving Chinese ink painting features and real-time rendering.

CHAPTER 2. REVIEW OF LITERATURE

This chapter focuses on the review of existing studies including Chinese ink painting, applications of non-photorealistic rendering (NPR), virtual painting, silhouette detection, and geometry-based NPR using 3D meshes. This chapter also introduces the basic concept of the neural network (NN).

2.1 Chinese Ink Painting (2D / 3D)

Chinese ink painting is a traditional painting style used in China. Artists use brushes to make strokes to substitute lines drawn with pens. The top image in Figure 2.1 shows a stroke painted with a brush and is typically used in traditional Chinese ink painting. The bottom image shows a simple pencil line and is mostly used for sketching and representing silhouettes and edges when drawing objects. In Figure 2.2, the left image shows a common mountain in the Chinese ink painting format. Only a general outline of the mountain is drawn by the artist. All details come from a process involving the bleeding of ink. The image on the right shows a mountain in a Chinese realistic painting style and involves a great number of lines not only at the edges but also in the shadow. Everything in the mountain scene was drawn by the artist.

Chinese ink painting art uses a special paper called “xuan” paper that allows the applied ink to bleed with ease. Left image of Figure 2.2 shows a real painting on a piece

of “xuan” paper. The image on the right is a simulation of an ink point bleeding on “xuan” paper. Thus artists don’t have to draw objects in detail. They just need to outline the main shape and a painted image will be developed by means of controlled color bleeding of the ink (Huang, Way, & Shih, 2003). M Sun, J Sun and Yun (2005) presented a controllable physical paper model to simulate the effect of Chinese Ink-Wash Drawing on “xuan” paper. Users can get different simulation results of the same input stroke by changing settings such as water-absorbability and density of that paper model (Sun, Sun, & Yun, 2005). Because of the inherent nature of “xuan” paper, the different characteristics of brushes, the different attributes of ink and even gravity, that can affect color bleeding, each Chinese ink painting product becomes truly unique.

But to apply this feature to real-time rendering, it’s nearly impossible to simulate every characteristic of the painting process. As a result, products generated from most studies turned out to be very different even though their internal logic was identical.



Figure 2.1 Examples of a stroke and line

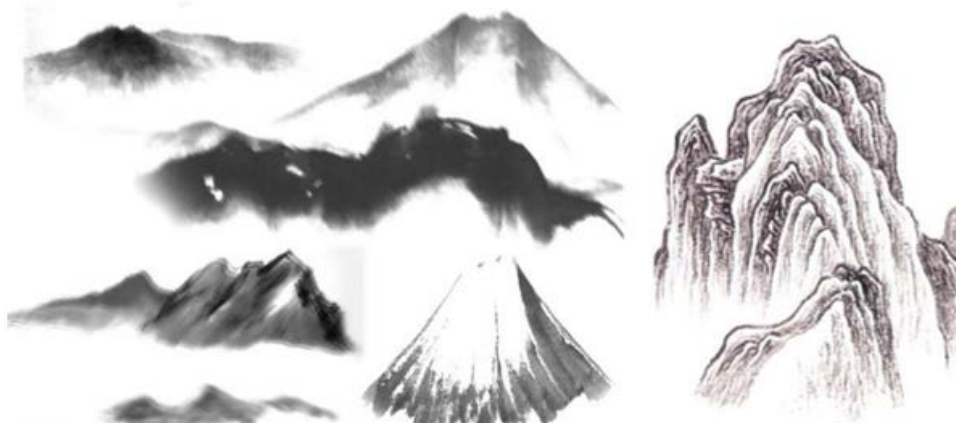


Figure 2.2 Mountains painted in different styles

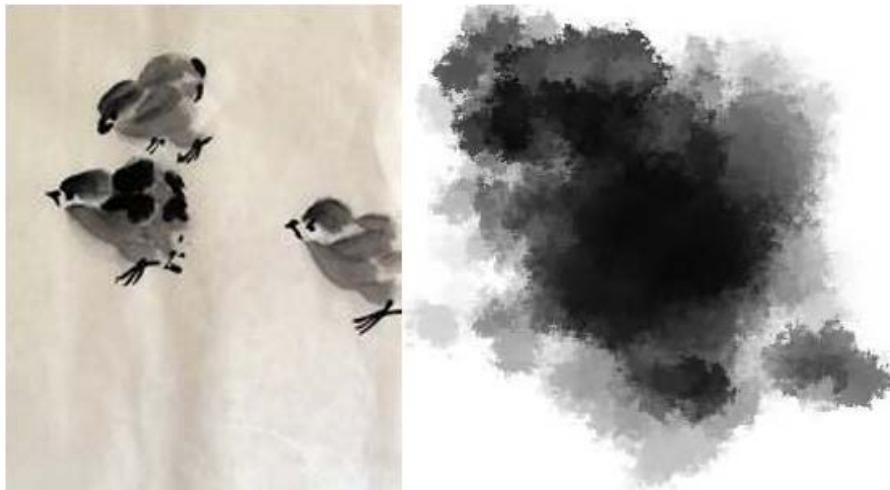


Figure 2.3 Example of ink bleeding

The development of Chinese ink painting in the industry of 2D rendering and animation is mature. Its use has already gained presence in a large number of creative designs and products. Unfortunately, their development was too time-consuming to be applied in 3D scenes. Also, the depth and extent of research addressing efficient methods for simulating brush strokes and the behavior of ink is small. As a result, such features seldom occur in the 3D object space even though the 3D rendering industry has developed quickly in the past years. In contrast to the large-scale deployment of graphics technology in western non-photorealistic rendering, little research has been done on the

technology for artists and programmers to efficiently create 3D scenes rendered with Chinese ink painting features.

2.2 Non-photorealistic Rendering (NPR)

To apply the Chinese ink painting to animations and games with changing motion and lighting, a technique called non-photorealistic rendering (NPR) is needed. NPR is a technique in computer graphics that applies a wide variety of styles in art to 2D or 3D scenes by rendering those scenes with approaches similar to painting steps.

There are some other painting styles presented in similar way as Chinese ink painting, such as oil painting. A lot of research projects focusing on rendering painting features in 2D or 3D scenes choose to use oil painting and acrylics (Isenberg, Neumann, Carpendale, Sousa, & Jorge, 2006). All painting styles including Chinese ink painting, oil painting, and others are important types of NPR. NPR also includes rendering cartoon, sketches and other features.

Some studies analyzed the understanding and assessment of hand-drawn pen-and-ink illustrations of objects in comparison with NPR renditions of the same 3D objects. Figure 2.3 shows a comparison between a hand-drawn and a computer-generated version of the same object. The image on the left is hand-drawn and the one on the right is computed by an NPR process. Comparative results show that people perceive differences between the two types of illustration because the computer-generated textures and silhouette look more organized (Isenberg et al., 2006).

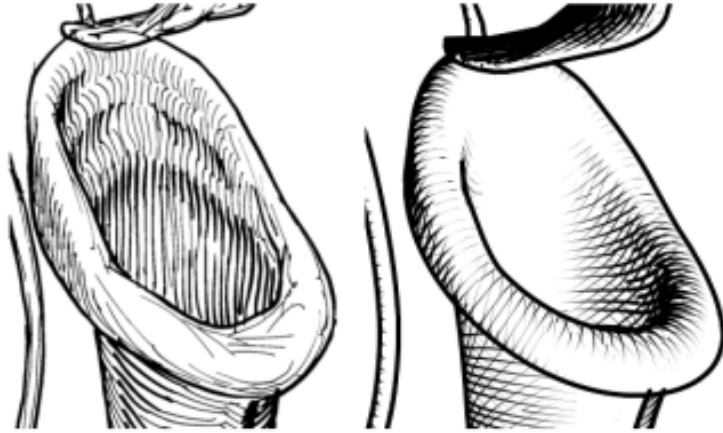


Figure 2.4 Comparison of real and NPR artworks (Isenberg et al., 2006)

Different research methods and features were noted in previous studies of NPR.

The left image in Figure 2.5 shows a cartoon version of NPR. It has a clear edge on every object and also has a color lump to fill the fields divided by the edges. The right image shows a sketched version of NPR. The entire scene consists of lines. Artists use pens of different colors to modify the color on objects. Figure 2.6 shows an oil painting version of an NPR product in which there are recognizable strokes. The scene was computer-generated with virtual brushes instead of pens. Baxter, Wendt and Lin (2004) developed a physical model for paint which is based on a conservative advection scheme that simulates the basic dynamics of paint. Ross (2000) presented an interactive system for computer aided generation of line art drawings to illustrate 3D models. Lum and Ma (2001) designed an approach to mimic the watercolor process by building up an illuminated scene through the compositing of several layers of semi-transparent paint. They also presented a method inspired by watercolor for the rendering of surfaces. Mitani, Suzuki, and Kimura (2002) proposed a new modeling and rendering system for displaying models in a sketch-like style. This system, called “3D SKETCH”, preserves the features of the user's strokes. And of course, the lighting models and rendering

techniques mentioned in the previous sections are all included in the system NPR (Baxter, Wendt, & Lin, 2004). Some Chinese researchers integrated the NPR technique with Chinese ink painting and designed algorithms for converting photorealistic images to a general ink painting style. Dong, Lu and Jin (2014) presented a novel real-time, automatic framework. This framework achieves a good performance of ink bleeding (Dong, Lu, & Jin, 2014).



Figure 2.5 Different NPR features (Cartoon and sketch)



Figure 2.6 Different NPR features (Oil painting)

The results of these studies show real-time rendering systems that automatically convert 3D triangle meshes into an appealing non-photorealistic style, or into 2D scenes (Mitani, Suzuki, & Kimura, 2002). The whole rendering process may include silhouette

extraction, or the transformation from points and edges to strokes and simulation of ink bleeding in shaders. Moreover, an appropriate lighting model can be generated for such 2D features in 3D scenes. Lighting elements will also be computed in interior shading. Sometimes the surfaces of an NPR mesh have attributes that reflect other objects. However, most studies paid great attention to light and shadow. Few show algorithms and models for non-photorealistic reflections. Because the ink painting process may be more time-consuming than line-art painting, developing an effective algorithm for rendering the bleeding of ink and color is important to improve the efficiency and performance of the result.

All of these studies concentrated on rendering non-photorealistic features into images to make them more similar to hand-drawn pictures. Previous studies always ended up by rendering non-photorealistic features into a 2D image or a geometry-based scene. Only a few of the studies took further steps to use the scenes generated with NPR to form more complex projects. Few studies applied NPR to real-time interactive projects such as games. There is some research that use NPR for real-time drawing. However, the efficiency of this approach found out to be more and more unacceptable as the number of painted elements increased. For example, the framerate for a loop of processes was sometimes turned to be even lower than 10 fps when generating more than 100 strokes. Performance with framerates lower than 10fps makes the experience of drawing unenjoyable for users.

2.3 Image-based Rendering

Research of NPR resulted in many findings that feature image-based rendering. This method uses image processing techniques; a brief description of these is presented here.

First an image would be imported. Then the characteristics of that image including silhouettes, color, lighting and so on would be analyzed. Information including depth, light source, and different objects would be collected for the rendering process. Finally, some parts of the input image or the entire image would be rendered with a new shading approach to achieve a different appearance.

Most research on simulating ink painting shows preference for the use of imaged-based rendering instead of geometry-based rendering. Encouraging progress was made in the past years in rendering scenes in ink painting styles. Danner and Winklhofer (1998) produced a cartoon with the appearance of a watercolor painting rendering out of a 3D scene (Danner & Winklhofer, 2008). Chu and Tai (2005) devised a novel fluid flow model to simulate ink dispersion on absorbent paper (Nelson Siu Hang Chu & Tai, 2005). Specifically for Chinese ink painting applications, Yang, Xu and Li (2011) proposed an image-based approach to simulate the painting procedure (Yang, Xu, & Li, 2011). Dong, Lu and Jin (2014) presented a real-time framework to automatically convert images into Chinese ink painting style. These are shown in Figure 2.7 (Dong et al., 2014). The left image is the input of their project. After detecting silhouette and simplifying details, the input image is converted to a Chinese ink painting product resulting in the image on the right. Their process discards color information and subtle texture while it enhances the background environment. Recognizable blur is added to simulate the appearance of fog. Based on these studies involving the rendering pictures in Chinese ink painting style, Guo, Peng and Tang (2015) present a new method to achieve it. The conversion procedure using their method has less user interaction and is much faster.



Figure 2.7 An example of real-time Chinese ink painting rendering algorithm (Dong et al., 2014)

A problem of implementing the process of image-based rendering in Chinese ink painting style is that different objects in the scene need to be rendered in a different way. But it's difficult to distinguish each object in an image, because computers can only interpret color information in each pixel. Even after detecting the silhouettes of objects, it is almost impossible to classify them. Another problem is that when applying the same algorithm to different objects, the style of the output may look very different because not all elements are appropriate for rendering in the Chinese ink painting style. This problem makes it difficult to unify the appearance of all elements in one image.

2.4 Virtual Painting

All of the existing NPR products apply their focus on the performance of different features. There are a great number of real-time drawing projects that were derived from previous efforts on NPR. Research on real-time drawing concentrates more on the properties of pen, brush and ink rather than the final appearance. Chen, Kim and Wang (2015) presented a real-time oil painting, using brush, paint, and canvas (Chen, Kim, & Wang, 2015). This system not only simulated the final paint effects but also the motion of

the brush and oil that occurred during the painting process. The approach noticeably improved the user experience.

For Chinese brushwork, this technique requires storing a large sample of stroke textures to avoid appearing repetitive. The strokes also appear unrealistic whenever there's a self-intersection or high curvature. The geometry of brushes was computed in some studies. In Chu and Tai's project (2004), skeleton and surface were constructed separately to make a reasonable brush, with skeleton handling the bend and dynamic, surface taking charge of flattening and spreading.

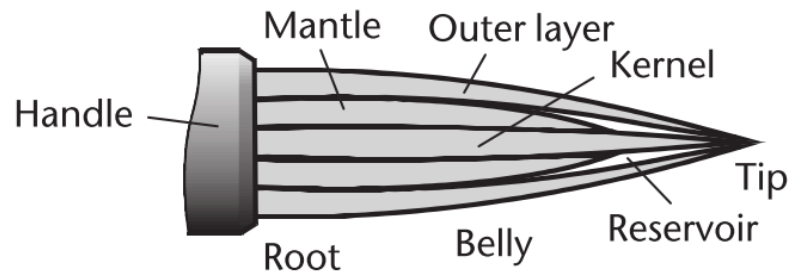


Figure 2.8 Anatomy of a typical brush (Nelson S H Chu & Tai, 2004)

Some physically-based models were generated to simulate properties including lighting and shading. At a more basic level, lighting can be divided into ambient, diffuse and specular (Huang, Way, & Shih, 2003, Lum et al., 2007, Ross, 2000). However, in Chinese ink painting, programmers should only take care of the ambient and diffuse terms of the lighting model because there is no smooth face that may require calculation of a specular term. Also, the ambient color is not the same everywhere anymore because it is supposed to mimic the gradient ramp on “xuan” paper.

Art products in Chinese ink painting style are always expressed in black and white instead of color. In computer graphics, a common method for representing paintings involves use of the RGB colors. The ambient, diffuse, and specular terms of

lighting on objects are computed separately as colors and then integrated to construct a complete lighting model. Baxter (2004)'s paint model, however, represented paintings in terms of paint pigments to relight paintings under any full-spectrum illuminant. Gooch, Gooch, Shirley and Cohen (1998) presented a model based on methods used in traditional technical illustration, where the lighting model uses both luminance and changes in hue to indicate surface orientation, while reserving extreme lights and darks for edge lines and highlights (Gooch, Gooch, Shirley, & Cohen, 1998). Kalnins et al. (2002) presented a system that lets a designer directly annotate a 3D model with strokes, imparting a personal aesthetic to the non-photorealistic rendering of the object.

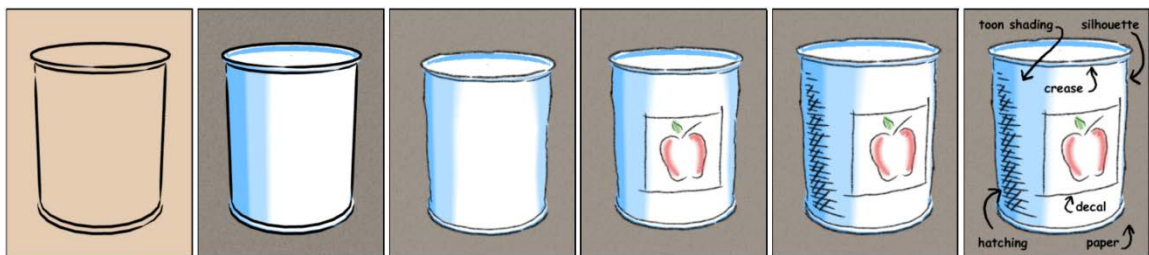


Figure 2.9 Example of the process of implementation of a real-time NPR drawing system (Kalnins et al., 2002)



Figure 2.10 Example of the details of a real-time NPR drawing system (Chen et al., 2015)

2.5 Silhouette Detection

Silhouettes play an essential role in NPR. Silhouette detection is the process of recognizing the outline of the objects in a 2D / 3D scene. Considering that silhouette detection on a 2D image is a latter part of detecting silhouette on a 3D mesh, this research only covered studies for 3D projects. Previous studies categorize silhouettes into five types: silhouette edges, sharp edge, valley edge, shadow outline and texture boundary outline (Wang, Tang, & Dong, 2004).

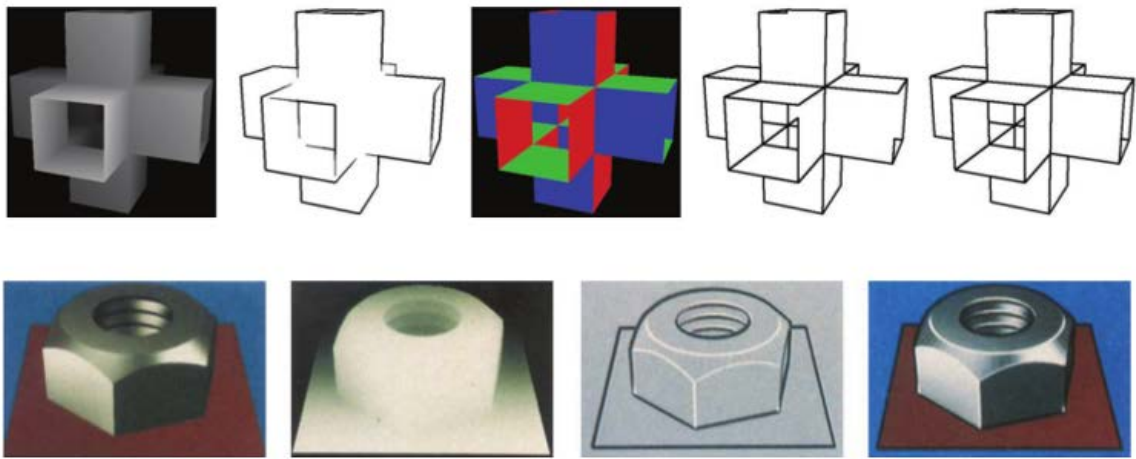


Figure 2.11 Basic silhouette detection methods (Hertzmann, 1999)

Three popular algorithms are used to detect the silhouette of a 3D scene. The first involves computing the dot product of the eye vector and the vertex normal. If these two vectors are perpendicular, then mark the vertex as a vertex on the silhouette. However, this approach may cause problems when the given meshes have concave faces. All edges of a front face and back face will be detected by this approach, which includes not only the boundary of the mesh but also redundant edges that constructed concave faces. For the NPR technique, sometimes those redundant edges may help simplify the process of generating new strokes.

Consider the top images of Figure 2.11. The third image shows the normal vectors of faces on a 3D mesh. The final data of silhouette would be the edges shown in the top right two images. But most of the time, the edges shown in the second image are required by the project. In such a situation, using the normal vector is not the best way to do silhouette detection.

The second algorithm involves the projection of 3D meshes to the image plane and then detecting the silhouette edges on the image plane. This type of algorithm is really effective. It can be done by detecting color continuousness in the color buffer. Silhouette colors can change abruptly. By applying this algorithm, the graphics hardware, existing libraries and packages help to generate buffers. The computational complexity of silhouette detection using this method does not depend on the number of polygons in the model anymore. Instead it depends on the pixels that comprise the image. Thus the image resolution remains constant because the number of pixels is defined (Wang et al., 2004).

The top left two images of Figure 2.11 show application of this algorithm. It also has the same shortcomings as the first algorithm. However, here we can correct for those by additionally rendering a depth image to check which edge belongs to the boundary and which edge is an “internal” edge.

A third approach involves rendering a depth map of the scene then detecting the silhouette of shapes on the depth map. By applying this approach, operators such as the Sobel operator are used to compute weights for each pixel to determine silhouette. As shown in the bottom left image in Figure 2.11, because the color on the depth map is easier to recognize than the color in a color buffer, it is possible to detect the edge faster

in this way than dealing with a colorful image directly (Hertzmann, 1999; Isenberg, Freudenberg, Halper, Schlechtweg, & Strothotte, 2003; Wang et al., 2004).

2.6 Pipeline of Geometry-based NPR

After silhouette detection, vertices not on the remaining edges can be discarded. In the rendering process, it's necessary to change edge lines to strokes, after which the strokes will be rendered to a texture. A multi-pass technique is important here to handle different render passes in real-time. Yuan, Yang, Xiao and Ren (2007) proposed a three pass rendering process which separates the tasks of interior shading, silhouette extracting, and background shading, see Figure 2.12.

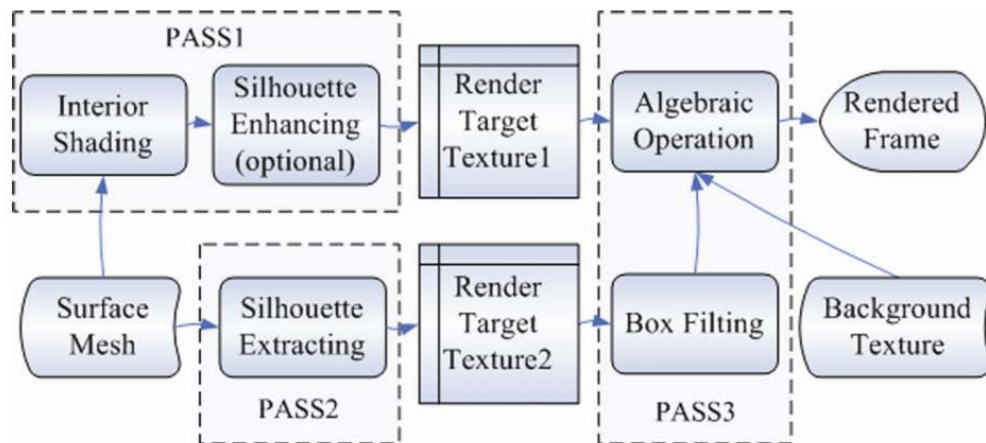


Figure 2.12 Sample of rendering process: the three passes of Yuan's whole rendering process (Yuan et al., 2007)

The process initializes two empty textures, one for storing the data of silhouette, the other for rendering the surface mesh and then preparing it for later use in shaders. The two textures are computed separately. After computation of textures in pass 1 and pass 2, the pipeline asks for integrating and applying the textures to the rectangle color buffer on the screen space.

Xu, Yang, and Wu (2012) present a rendering system for a real-time ink wash painting style rendering of geometric models. They proposed a PADI structure to deal with the color buffer that involves simultaneously integrating the occlusion culling, silhouette extraction, and shading in the first pass to get the basic performance of the final output. Resolution requirements and anti-aliasing are dealt with in the second pass. They also add a geometry shader in the second pass to implement a particle system at each point to simulate strokes. In the project made by Xu, Yang, and Wu, the simulation of an art effect called “fly-white” improves the performance significantly (Xu, Yang, & Wu, 2012).

2.7 Neural Network (NN)

Many of the techniques applied in this research project are similar to a neural network. A neural network is a popular concept in machine learning. It can be described as an information processing paradigm inspired by the biological nervous systems (Nahar, 2012; Taormina, Chau, & Sethi, 2012). There are different types of neural networks. These include deep neural networks (DNN), convolutional neural networks (CNN), feed-forward neural networks (FNN), and so on (Gregor, Danihelka, Graves, Jimenez Rezende, & Wierstra, 2015; Ngene, Agunwamba, Nwachukwu, & Okoro, 2015; Taormina et al., 2012).

A neural network requires several input nodes. The input nodes do not recognize and modify the data they store. The entire neural network receives simplified values from input nodes, processes unknown activation functions using the values, and gets multiple outputs. The main task of these network is to determine the process occurring in the

hidden layer that is between the input and the output layers. A schematic of a neural network is shown in Figure 2.13.

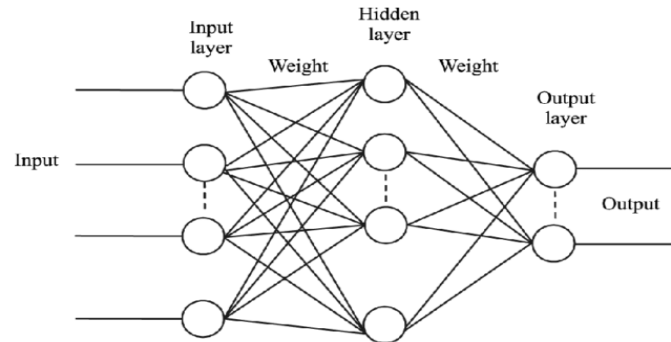


Figure 2.13 A general structure of an artificial neural network

Sometimes the procedure in the hidden layer can be a Gaussian convolution with smooth step techniques when the inputs and outputs are dealing with colors. The Gaussian function can combine and compute the weight of the color in each pixel. The smooth step function is a nonlinear function used to compute the final result for each output nodes. As deep learning progress in a neural network is made, and if output nodes have grey colors closer to adjacent pixels than input nodes, or if the output colors are gradually changed between adjacent pixels, then the network can determine out whether the process in the hidden layer is a blur function followed by a smooth step or not. Gaussian convolution and smooth step techniques work as activation functions and perform the same function as the hidden layer in a neural network (Deng, Hinton, & Kingsbury, 2013; Kalchbrenner, Grefenstette, & Blunsom, 2014).

In later sections we compare the NPR technique we implement to the transformation a NN applies to its input data.

2.8 Summary

This chapter presents the analysis of projects in the field of Chinese ink painting and NPR. A gap can be found that there are few researches on applying ink painting features to interactive scenes.

With the respect of the efficiency of previous projects, mostly when dealing with simple elements such as vertices, edges and faces, their programs work much faster than computing and rendering strokes. If particle systems are used to simulate the stroke or ink bleeding, then the process works even slower. With respect to performance, strokes and bleeding color are more similar to hand-drawn art than computed line-art features. That is because machine-computed lines are being generated based on internal rules which makes them lack a random look. However, the randomness of the bleeding color process is harder for people to differentiate from hand-drawn art products. The next chapter will discuss the methodology used in developing the research project in this study.

CHAPTER 3. METHODOLOGY

This chapter introduces the framework of developing a real-time rendering project. It includes a description covering application of a general Chinese ink painting style, the design of different data structures, the computing tools used, the algorithm to be applied, and the rendering pipelines for static and moving objects. This chapter also describes the testing procedure and the assessment instruments used.

3.1 Development Tools

This research project will be developed using OpenGL, OpenGL Shading Language (GLSL). The models will be made in Autodesk Maya and 3ds Max. The format of input meshes will be “.obj”, “.ply” and “.dae”.

3.2 Framework

The purpose of this research is to determine an efficient approach to render Chinese ink painting features in 3D scenes. Before introducing the technical portion of the appearance used, the framework of the project shall be described.

In this project, rendering of a typical scene shall consist of three main elements. See Figure 3.1. The first involves moving objects which are defined as objects with some basic artificial intelligence and animations. Users can interact with them to trigger the animations. The second includes static objects, defined as motionless items and can be looked at in different directions. However, no transformations will be performed on such

items. The last element represents the background in a scene and which can include fog, cloud, a landscape or some distant geometries.

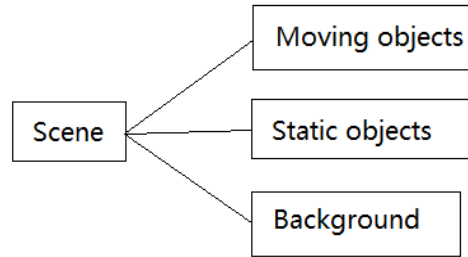


Figure 3.1 Elements to be rendered in a scene

From an art standpoint, objects in Chinese ink painting style consist of brush strokes instead of points and line edges. The 2D ink painted objects lack a clean and distinct silhouette and the lighting is not well-distributed on each fragment. Figure 3.2 presents the framework of the overall design process used in this study. The elements shown serve to provide a logical basis that will lead to a successful design. Also, this research has to address a method to eliminate most redundant edges on the original objects and then do the conversion from core edges to brush strokes. After simplifying the objects, some desired lighting environment has to be generated for the whole scene. When objects move, the bleeding of color will not remain the same as that for objects that are static. Also, there has to be performance of flow as well as changes involving the use of shade.

Besides rendering, the available control logic is also needed to be designed for users to interact with interactive objects in the scene.

To achieve the desired level of art quality and aesthetics, the designer must make use of appropriate algorithms that include silhouette detection, different rendering passes,

Gaussian blur, smooth step functions, and a particle system for the simulation of stroke-like appearance. Interior shading for each object is computed based on the designed lighting model and the Gaussian blur will be applied to simulate the effect of ink bleeding.

Moving objects and static objects will be rendered in different ways. Moving objects need a structure to store mesh skeletons for animation. The interior of moving objects will be simplified for the case where the diffuse color turns out to be unacceptable as often happens when objects are moving or deforming. For ink simulation, the shading will include more details of a gradient ramp of diffuse color as well as some art features such as “fly-white” or ink-bleeding

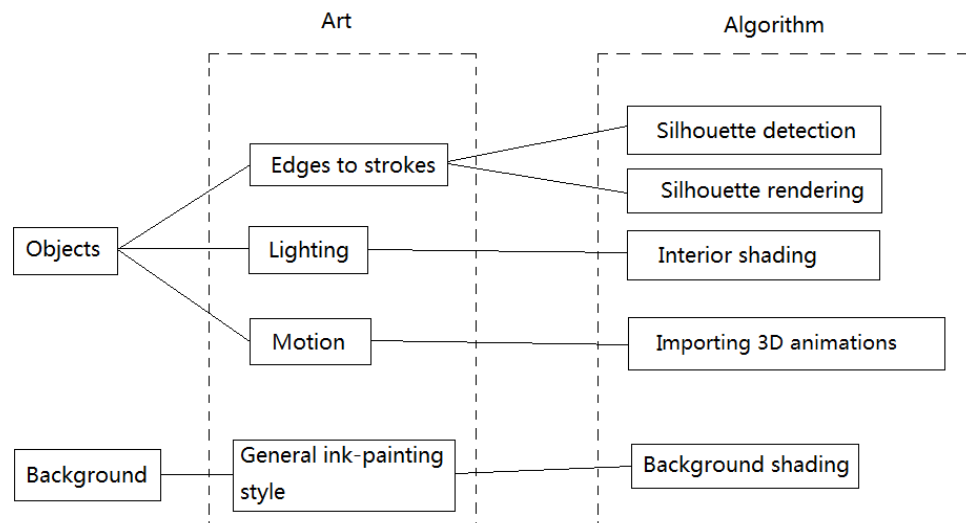


Figure 3.2 Design of the proposed project

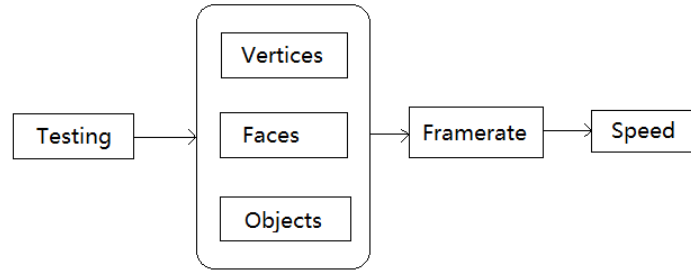


Figure 3.3 Testing of the proposed project

3.3 Rendering Pipeline

3.3.1 Pipeline for rendering static objects

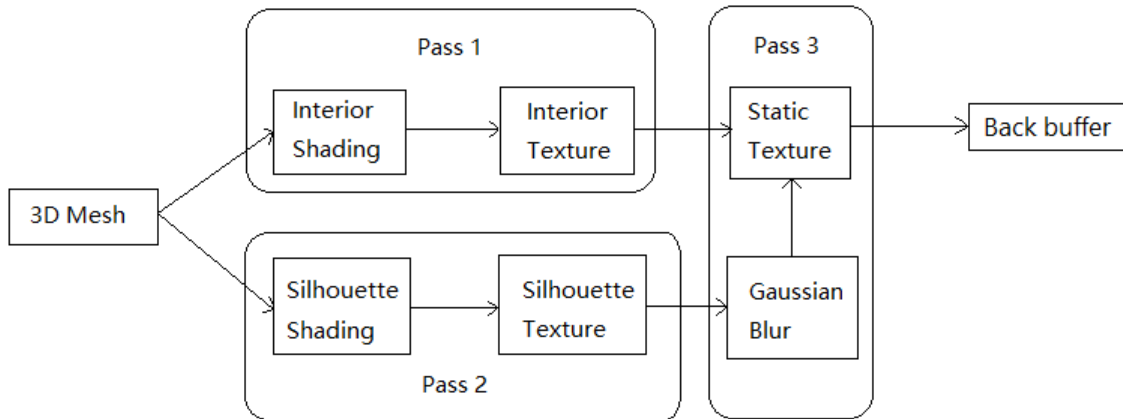


Figure 3.4 Pipeline for rendering static objects

In this research, the non-photorealistic style of a static object was defined as a kind of Chinese ink painting style. Figure 3.4 shows a schematic of the pipeline for rendering static objects achieve the static object target feature in real-time, the rendering process shall use the following steps:

1. Initialize the two textures and framebuffer objects; one for the interior, the other for the silhouette.

2. Render the 3D mesh to the first texture. Apply interior shading to this rendering pass.
3. Render the 3D mesh to the second texture. Apply silhouette shading to this rendering pass.
4. Apply a 2D Gaussian blur to the second texture using the Gaussian

function: $G(x,y) = \frac{1}{4\pi} e^{-\frac{x^2+y^2}{4}}$. Integrate the two textures to get the final color of each fragment. Draw a rectangle whose size fits the window perfectly. Render the rectangle to the screen with the final color.

3.3.2 Pipeline for rendering moving objects

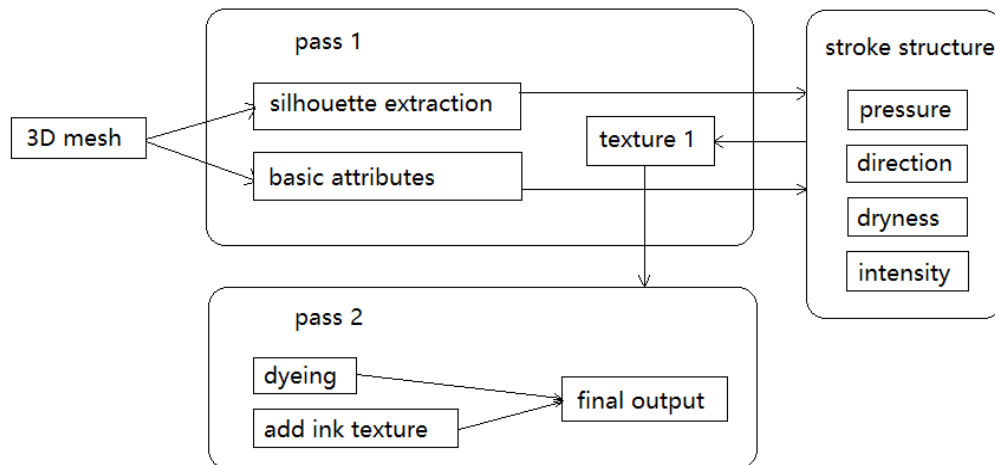


Figure 3.5 Pipeline for rendering moving objects

To render moving objects, a stroke structure is designed to save the information of a stroke as shown in Figure 3.5. The structure has four variables namely pressure, direction, dryness and intensity. Pressure indicates the width of the stroke. Direction is a unit vector parallel to the stroke. Dryness is used to simulate the “fly-white” feature. The value of dryness will be randomly assigned. When dryness is less than some pre-defined

constant value, that part will not be dyed. Intensity is a coefficient of the ink color.

Strokes with higher intensity will be darker.

The total rendering for moving objects has two passes. In the first pass, the vertex attributes, including position attributes, normal attributes and texture coordinate attributes, will be passed to shaders. A depth buffer will also be rendered for occlusion culling.

The vertex shader simply converts vertex data to a world coordinate system and then passes the data to the next stage. Silhouette strokes will be detected and extracted in the geometry shader. For each triangle, if the normal vector of one of its three vertices v_0 is different from the others namely v_1 and v_2 , then the triangle is on the silhouette field. The equation $[(1 - \lambda)\vec{N}_0 + \lambda\vec{N}_1] \cdot [(1 - \lambda)\vec{V}_0 + \lambda\vec{V}_1]$ is solved to determine the normal vectors based on the normal attributes of v_0 , v_1 and v_2 for all points inside the triangle. A segment along a path where the normal vector equals 0 inside the triangle is then created. The new segment is a silhouette segment. Considering the stroke structure, the pressure value will be set depending on the length of this segment. It is bigger at the center of the segment than at the ends. By setting the direction variable, the ink stroke will be parallel to the silhouette segment and go clockwise.

New geometry is created using two ends of a silhouette segment as the basis. Adding two vertices above the segment and two vertices below it will form a new rectangle. The width of the new rectangle depends on the pressure value. Those four new vertices will have texture coordinates (0.0, 0.0), (0.0, 1.0), (1.0, 0.0) and (1.0, 1.0).

Stroke information and texture from the first pass are analyzed. Each new geometry will be attached by an ink texture in the fragment shader. The ink texture

consists of particles that are generated in the geometry shader. The complete pipeline is shown in Figure 3.5.

3.3.3 Interior Shading

Interior shading is done with the following steps:

Passing the position and normal attributes of each vertex from the vertex shader to the fragment shader.

In the fragment shader, compute the light vector, then determine the diffuse attribute by computing the dot product of the normal vector and the light vector:

$$\text{diffuse} = \vec{N} \cdot \vec{L}.$$

The color of each pixel only depends on the diffuse variable in the interior shading. When the value of $\text{diffuse} \leq 0.25$, the output color needs to be multiplied by 1.0. When $0.25 < \text{diffuse} \leq 0.55$, the output color needs to be multiplied by 0.7. When $0.55 < \text{diffuse} \leq 0.8$, the output color needs to be multiplied by 0.4. When $\text{diffuse} > 0.8$, the output color needs to be multiplied by 0.1.

Moreover, to simulate the ink bleeding feature on the target texture, a step of 1D Gaussian blur needs to be applied to the output color in the fragment shader. The

Gaussian function used here is $G(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$. Figure 3.6 shows the comparison

between the original shading and the shading after Gaussian blur.



Figure 3.6 1D Gaussian blur

3.3.4 Silhouette Shading

Silhouette shading is done using the following steps:

Transferring the position, normal attributes and texture coordinates of each vertex from the vertex shader to the fragment shader.

In the fragment shader, compute the view vector. Silhouette is the outline of the mesh and is view dependent. Those fragments where the normal vector is perpendicular to the view vector are on the silhouette. Thus, a variable called “edge” was defined in this project to compute the dot product of the normal vector and the view vector: $\text{edge} = \vec{N} \cdot \vec{V}$.

Another variable called “threshold” was defined to set the width of the silhouette. When $0 < \text{edge} < \text{threshold}$, the fragment needs to be shaded. The same 1D Gaussian function that is used to define Gaussian blur in Figure 3.6 is applied again to make the color of silhouette change gradually from black to white. By so doing, the silhouette texture is able to merge with the interior texture with no gap between the silhouette part and the interior part.

Sometimes ink strokes are not solid everywhere because these can be influenced by brush materials, drawing pressure, paper material, gravity, and so on. Three noise textures, shown in Figure 3.7, are used to simulate ink texture.

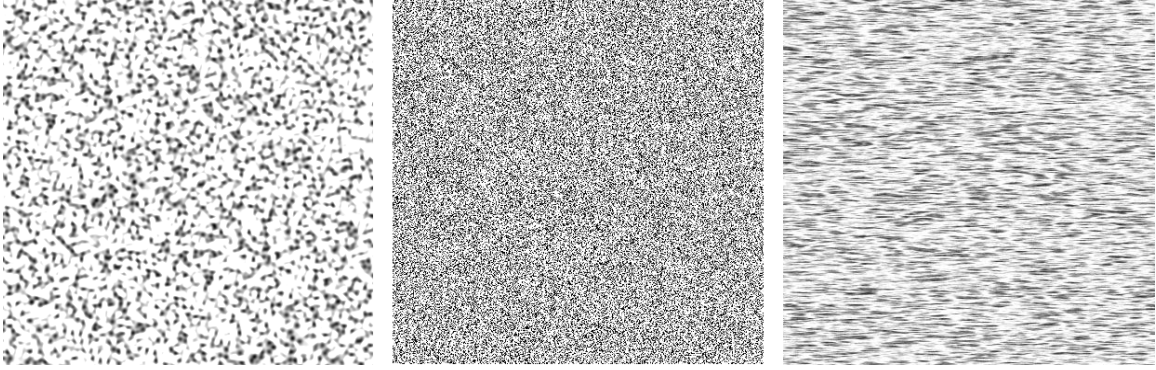


Figure 3.7 Noise textures

The left image in Figure 3.7 represents simplex noise that can be directly generated in the fragment shader. The middle image and the one on the right are two additive noise textures that can be attached to the silhouette strokes by using texture coordinates.

3.3.5 Static Texture

Before integrating the interior and the silhouette textures, a 2D Gaussian blur step is needed for the silhouette texture in order to get the anti-aliasing result. The 2D Gaussian function is $G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$. In the equation, σ^2 is affected by the Gaussian kernel width. To optimize the speed, this Gaussian convolution can be implemented as a horizontal followed by a vertical blur.

After integrating the interior texture and the silhouette texture into the final static texture, a built-in smooth step function of GLSL is applied in the fragment shader. The purpose of the smooth step function is to perform a Hermite interpolation between the values of the first two parameters in that function to determine the third parameter. The plot in Figure 3.8 shows how the smoothstep() function is used. In this research project, the parameter x represents the grey color variable values in the range from 0.0 to 1.0. When the color value is less than Edge0, then modify it to 0.0. When it is more than

Edge1, then modify it to 1.0. By using this approach, the attribute of smooth shading and a clear silhouette will be realized.

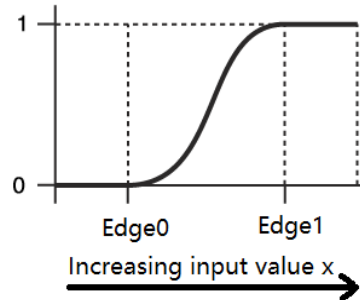


Figure 3.8 The GLSL function $\text{smoothstep}(\text{Edge0}, \text{Edge1}, x)$

Gaussian blur and smooth step techniques used in this research project can be regarded as an implementation of a neural network instead of a deep learning process. The Gaussian blur process takes each pixel in the Gaussian kernel as an input node. It only receives the grey color information from each pixel. The Gaussian function can combine and compute the weight of the color in each pixel. The smooth step function is a nonlinearity function used to compute the final result of each output node. The two functions can be regarded as activation functions that perform the same task as the process in the hidden layer of the neural network. The technique used in the research project is similar to that in a convolutional neural network with features being selected rather than learned.

3.3.6 Skeleton Animation

Besides the basic mesh structure, which only contains vertex attributes, a new data structure is needed to load the skeleton of meshes into the rendering project.

When modeling the meshes in this research project, one vertex can be affected by as many as 8 bones. A hash table is used for storing the relations between one vertex and

the bones that will affect this vertex. The table stores the bone IDs and weights of each bone. The data is then passed into the shaders as several vec4 variables. For each bone, a data structure is designed to store its offset from the corresponding vertex's transform and its final transform.

For moving objects with skeleton animation, the position attribute will no longer come from the vertex directly, but is computed by calculating the sum of the related bones' positions multiplied by the bones' weights.

3.4 Assessment Instruments

The speed and quality of the scene are then analyzed. A table including the complexity of meshes and the frame rates needed to render each mesh will be generated. This research project is targeting a framerate of more than 60 fps for rendering an entire scene having less than 1,000,000 vertices.

3.5 Summary

This chapter provides an overview of the design and development process used to render 3D scenes with a Chinese ink painting style. Successful completion of this research project needs to have an acceptable efficiency and rendering techniques need to be available to be embedded in an interactive scene or a small game in Chinese ink painting style.

CHAPTER 4. IMPLEMENTATION

This chapter displays the details of developing the thesis project, including each independent feature and the process of combining them. It shows the comparison of different features when changing exact parameters. This chapter also displays the implementation of some techniques not been used in the final result. And it gives the reason for discarding them.

4.1 Interior and Silhouette Shading

4.1.1 Basic Interior and Silhouette Shading

The left image in Figure 4.1 shows the interior part of the meshes that are rendered to a texture in the first render pass. To enhance the silhouette of the meshes, silhouettes are detected and rendered to another texture in the second render pass with different features generated in the fragment shader. The results of the rendering process are shown in the right image of Figure 4.1.

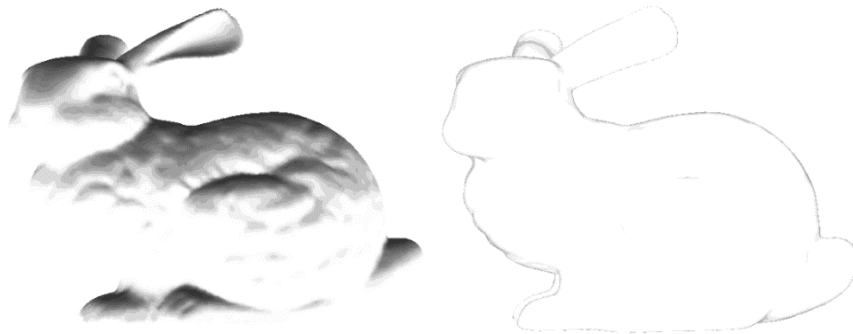


Figure 4.1 Performance of basic interior rendering and silhouette rendering

By modifying “threshold”, a variable defined in 3.3.4, the width and intensity of the silhouette stroke will change. To find the best performance for each mesh, different “threshold” values are passed to the fragment shader. Figure 4.2 shows a comparison of the features with different “threshold” values. The “threshold” value of the bunny silhouette in the left image is set to 0.2. The middle image shows the bunny silhouette with a “threshold” value 0.5 and that of the image at the right is 0.8. The bunny mesh that results in the best looking ink painting style is obtained when the “threshold” is set to 0.5. Therefore, when illustrating other features in the following sections, the “threshold” value will always be set to 0.5. But in the whole project, each mesh has its own values for each variable in order to get the best performance.

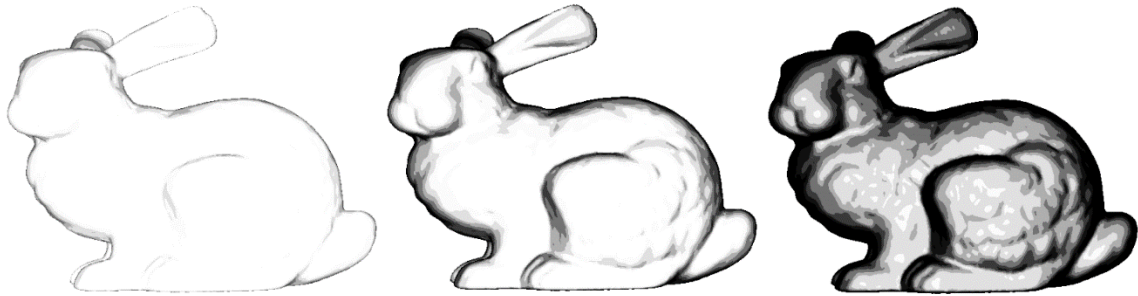


Figure 4.2 Comparison of features with different “threshold” value

4.1.2 Gaussian Blur

Meshes with basic rendering still have aliasing silhouettes. Moreover, though the interior shading gives a good simulation of color bleeding in the interior parts of meshes, there is no color bleeding feature outside the meshes. To solve this problem, a 2D Gaussian blur is applied to the texture where rendering of the silhouette is made.

Figure 4.3 shows the performance of silhouette texture rendering with different values of the Gaussian kernel. Here the three bunny renderings have kernel width values of 3, 7 and 11, respectively. When the kernel width is too small, the level of blurring is

not enough to provide for anti-aliasing. In the left image of Figure 4.3, aliasing appears especially on the edges of the ear part of the bunny mesh. Also, when the kernel is too big, the meshes will appear fuzzy. In the right image of Figure 4.3, the bottom of the mesh looks really fuzzy, resulting in an image that becomes similar to that being displayed on a screen with low resolution. Therefore, for the bunny mesh, 7 is an appropriate kernel width that both meets the anti-aliasing goal and proper simulation of color bleeding of the silhouette stroke. In this research project, the silhouette of all meshes in the final scene will apply a Gaussian blur with a kernel width of 7 pixels.

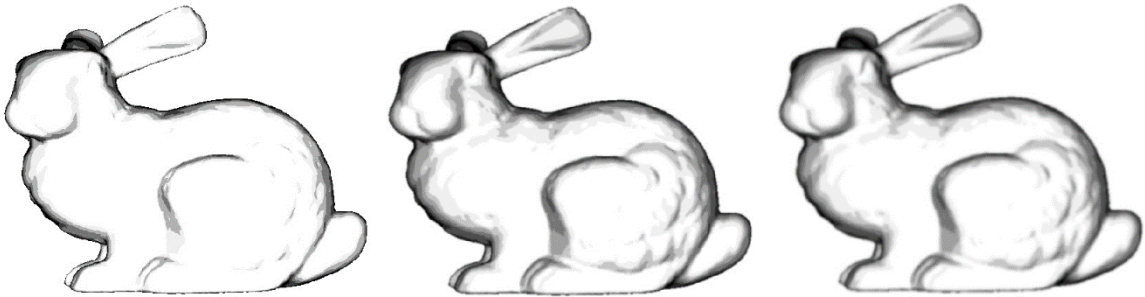


Figure 4.3 Silhouette Gaussian blur with different kernel width

4.1.3 Smooth Step Function

Refer to Figure 3.8. By applying the built-in smoothstep (genType edge0, genType edge1, genType x) function of GLSL to the final texture that integrates both interior and silhouette textures, the ink intensity will be strengthened at the dark parts of the silhouette strokes. The bound of color lumps which is adjacent to white (1.0, 1.0, 1.0, 1.0) and black (0.0, 0.0, 0.0, 1.0) will be clear and identifiable rather than fuzzy. It will also not degrade the anti-aliasing performance.

Figure 4.4 shows the final performance after integrating both interior shading and silhouette shading. The rendering in the left image has no smooth step function being

applied. In the middle image, parameter “Edge0” is 0 and “Edge1” is 1. It only smoothes the colors from white to black. Rendering in the right image uses a smooth step function with “Edge0” 0.1 and “Edge1” 0.92, resulting in the best performance involving a Chinese ink painting style.

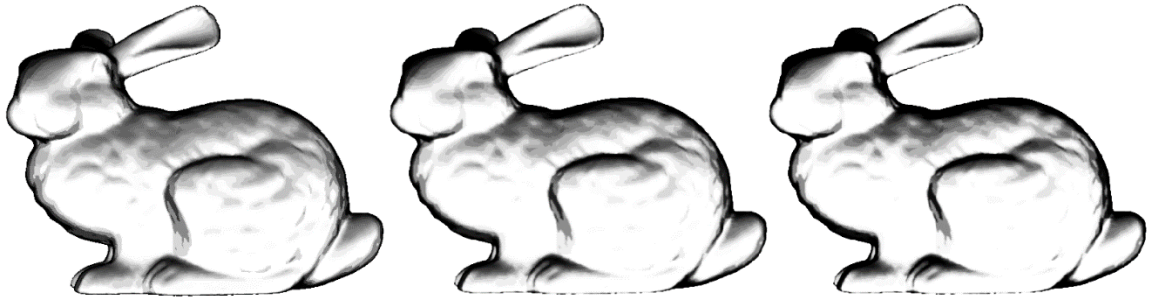


Figure 4.4 Effect of varying smoothstep edge parameters

4.1.4 Ink Texture

Figures in the previous sections show different renderings with all silhouette strokes dyeing by solid ink color. In the implementation of this research project, experiments are conducted using different noise textures to determine which ones result in the best looking Chinese ink painting style.

As mentioned in chapter 3, three noise textures are used to simulate ink stroke.

Figure 4.5 shows their results in a scene involving a stone mesh.

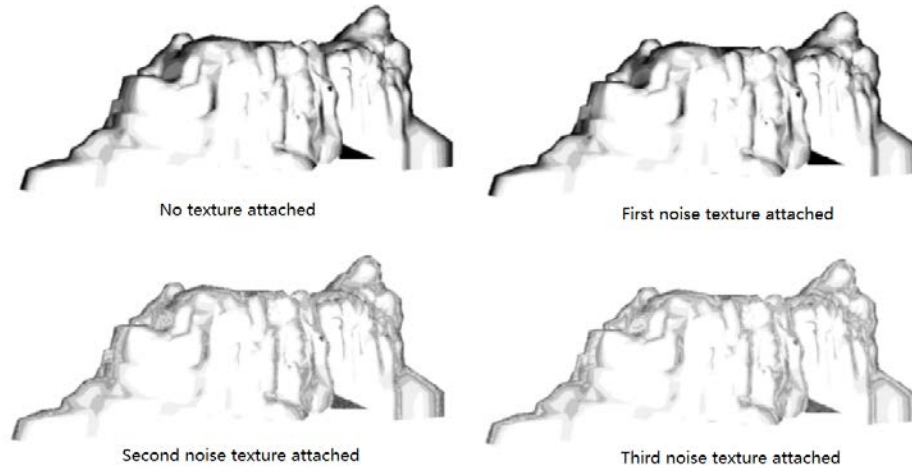


Figure 4.5 Silhouette with different noise textures attached

The simplex noise texture, presented in the first image of Figure 3.7, takes time to compute. But when applied to the silhouette, it is not identifiable enough as shown in the top-right image in Figure 4.5. Because of high cost, it will not be used in the final scene rendered by this research project. The two additive noise textures, presented by the middle and right images of Figure 3.7, create good shapes for the mesh silhouette as shown in the bottom two images in Figure 4.5. However, they look more like art products drawn with pencil rather than ink and brush. Therefore, they will not be used to render the final scene.

The best performance as Chinese ink painting style is still created the same way as demonstrated in the last section, namely, by setting the value of “threshold” to 0.5, Gaussian kernel to 7, and smooth step from 0.1 to 0.92. Additional experiments for finding a good texture to simulate ink strokes can be made.

4.2 Other Experiments

Besides the features chosen in previous sections, there are implementations of other methods mentioned in chapter 3. However, because the output of those experiments is of

a poor quality, or because the output is not similar to the desired Chinese ink painting style, those results will not be used in the final scene.

4.2.1 Stroke Structure

As illustrated in section 3.3.2, the implementation of generating silhouette segments works well for meshes with no cusp vertices. The left image in Figure 4.6 shows the silhouette segments of a teapot mesh. Some parts of the segments disappear because of the resolution. The original picture with higher resolution can be found in Appendix A. The method presented in section 3.3.2 detects not only the outside silhouettes of meshes but also the inside contours. After processing in the geometry shader, new geometries that provide the appearance of strokes are created. Those new geometries change the width of the silhouette. The middle image in Figure 4.6 displays the result of silhouette strokes before converting to the defined stroke structure. Each new geometry has its own texture coordinate and contains randomly generated particles in its attached texture as the stroke structures shown. The right image in Figure 4.6 shows the final result that features ink texture and changing stroke width. Each geometry has a different value of pressure variable depending on the length of the corresponding segment. A value of dryness variable was randomly assigned to each geometry. Discontinuous parts on silhouette strokes appear where the value of dryness variable is less than 0.3. This is to simulate the “fly-white” feature.

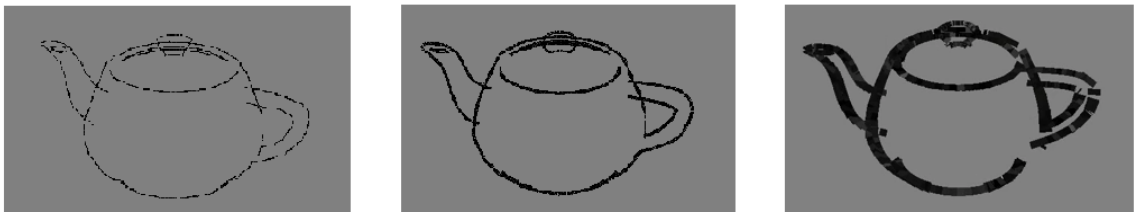


Figure 4.6 Stroke structure implementation

The final output image of this implementation is far from being regarded as ink painting style. Although the ink texture looks good, the “fly-white” feature made by this project is totally different from real art products that have “fly-white”. Also the output has no color bleeding in the interior part of objects. Therefore, the pipeline designed for moving objects will not be adopted in the final scene.

4.2.2 Watercolor Style

Another edge detection method was addressed to produce a different mesh effect. Pixels belonging to silhouettes are separately computed in a horizontal and vertical direction. Edges have a simplex noise texture that makes it possible to change their strength in places along the silhouette. Then the smooth step function was applied. It helped to smooth the edge color. The interior of meshes were dyed using the dot product of texture color and grey (0.3, 0.3, 0.3, 1.0). The interior was also affected by the transparency of each fragment. The left image of Figure 4.7 shows the mesh with the original texture. The right image is the output image of this rendering pipeline.



Figure 4.7 Watercolor style rendering

By implementing this method, the original texture of meshes can be kept. Objects may have more details than applying the methods addressed in previous sections. Because the right image of Figure 4.7 looks closer to a watercolor style than to the designed ink painting style, it was discarded from the final scene. Chinese ink painting style does not pay attention to details such as textures. Chinese ink painting is a genre of

“free hand brush work” instead of “true-life” artwork and always focuses on rendering a misty environment. Briefly stated, main elements in hand-drawn Chinese ink painting products are diffuse color, shadow and strokes.

4.3 Summary

This chapter demonstrated the implementations and other rendering styles not adopted in the final scene. It describes the steps taken and the progress made to find the best output style for a Chinese ink painting. The rendering pipeline for static objects was explained in detail. The chapter includes the comparisons of performance when different values of each defined variable were used. This chapter also displayed the results of other outputs generated by methods such as the rendering pipeline described in 3.3.2 and a new algorithm for silhouette detection.

CHAPTER 5. RESULTS

This chapter presents the output images of rendering involving single meshes and big scenes. It discusses the speed of the process used in the rendering pipeline by computing the framerate of rendering scenes while varying the number of vertices. Outcomes are analyzed to see whether the rendering technique is real-time or not.

5.1 Rendering

5.1.1 Single Meshes

Although Chinese ink painting generally contains pine trees, mountains, and fog, in this research project, bamboos, pavilions and cranes are modeled and rendered to determine the performance of the single mesh rendering process. Figure 5.1 shows the view of three original models in Autodesk Maya. Figure 5.2 shows the output of rendering those models in a Chinese ink painting style format.

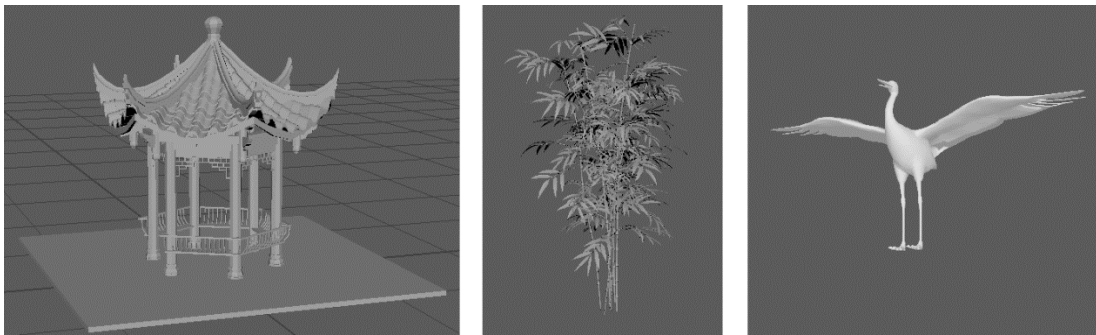


Figure 5.1 Original models



Figure 5.2 Models rendered with the pipeline proposed by this project

5.1.2 Final Scene



Figure 5.3 Final static scene

Figure 5.3 shows the final result of rendering static objects. The background image is a hand-drawn Chinese ink painting artwork. The close-up objects are rendered

with the pipeline proposed by this research. Static objects include the terrain, cranes, mountains, stones, bamboos and a pavilion.

5.2 Testing

The hardware used for testing the result of this research project is listed below:

Processor: 2.50 GHz Intel Core i7 – 4710 HQ

Display: NVIDIA GeForce GTX 970M

Table 1 Speed testing

Scene / Mesh	Number of vertices	Number of faces	Framerate
Bunny	35,947	69,451	48
Stone	2,084	4,060	>60
Crane	2,221	4,438	>60
Pavilion	10,840	18,996	50
Terrain	34,177	33,774	47
Bamboo	86,103	149,433	50
Static scene	467,632	423,554	36
Final scene	1,141,076	959,274	30

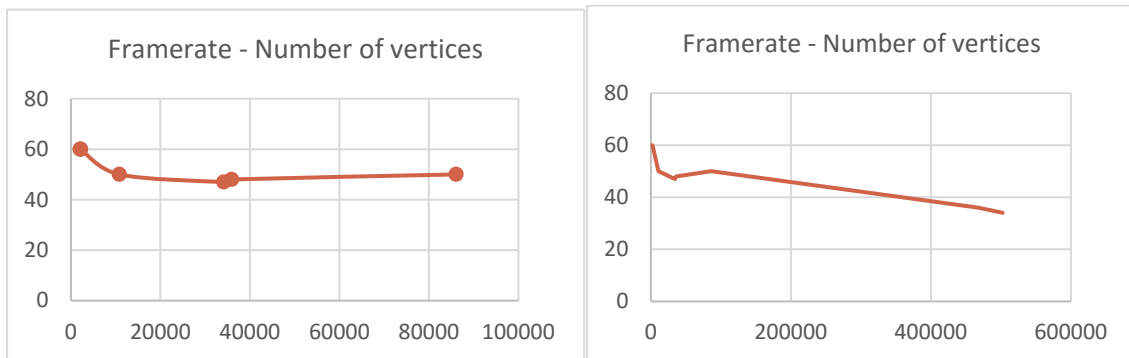


Figure 5.4 Speed testing with different number of vertices

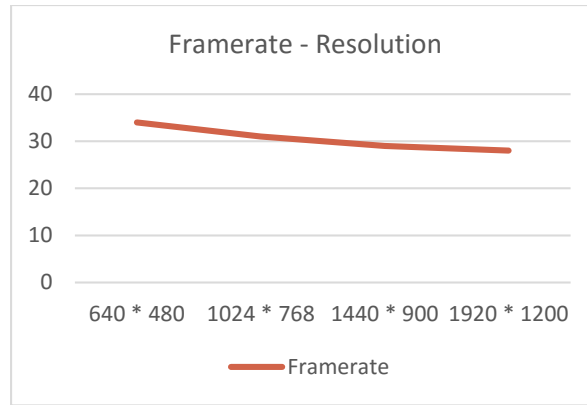


Figure 5.5 Speeding testing on screens with different resolution

The initializing time of the final scene is about 3 seconds and it increases as when the number of meshes increases. The main cost is for loading meshes and initiating OpenGL settings. The resolution of the testing screen is 1000 * 1000. The results in Table 1 show that the framerate of rendering a single mesh with Chinese ink painting features is far higher than 30fps. See Figure 5.4. As the number of vertices in the entire scene increases, the framerate is steadily reduced. The more vertices added to a scene, the slower the framerate turns to be. When the number of vertices is greater than 500,000, the framerate remains higher than 30fps. When the number reaches 1,141,076, the framerate is 30fps. The final scene that has 1,141,076 vertices was also rendered in screens with different resolutions. The testing result is shown in Figure 5.5. In screens with commonly seen resolutions including 1024 * 768, 1440 * 900, the framerate is about 30fps. When the resolution reaches the full HD level (1920 * 1200), the program still runs with 28fps. The speed is still acceptable as for a real-time program. Thus, it is reasonable to make a hypothesis that the rendering pipeline can work with framerates higher than 30fps when the entire scene has less than 1,000,000 vertices in most devices. Even in some devices that have higher resolution, the framerate will not be less than 24fps. At this level, the rendering pipeline is considered to function in real-time.

5.3 Summary

This chapter displayed the results of this research project. Examples of rendering single meshes and big scenes were presented. The speed of rendering scenes with different number of vertices were evaluated. The data showed that for most scenes with less than 1,000,000 vertices, the framerate is higher than 30fps. As a result, the rendering given by this project can be regarded as “real-time”.

CHAPTER 6. CONCLUSIONS

The goal of this research is about rendering 3D meshes in a Chinese ink painting style. This research project focuses mainly on the gap between the existing and the integrated methodology used. This gap was narrowed as indicated by the results shown. By so doing, one can conclude that the main objective of the research study was met even though other methods for generating quality Chinese ink painting rendering are available and should be explored.

6.1 Summary of the Research

In detail, the research project is an implementation of several NPR methods including edge detection, interior and silhouette shading, ink texture simulation and special lighting model for ink painting style. The final result is an interactive scene where all objects are rendered in a more realistic and authentic Chinese ink painting style. The scene contains background texture, static objects and moving objects. This project is also a frame for rendering other models as long as the models have acceptable format. The framerate of the rendering pipeline proposed by this research is higher than 30fps and is considered to be a real-time rendering.

6.2 Future Work

Future work on Chinese ink painting rendering should first be aimed at adding other features and details to the scene. A method for simulating the “fly-white” feature was mentioned in chapter 4. Although it was eliminated from this project because of poor

performance, “fly-white” is a special and interesting feature in Chinese ink painting style. It is good to design a suitable algorithm in the future to simulate it. Also, Chinese ink painting still has a great number of other features including “wrinkle texturing”, “brush tip strengthens” to be simulated in doing geometry-based rendering.

The results of this research project are presented in black and white. New methods can be addressed for dyeing the output scenes with different colors.

In real Chinese ink painting products drawn by artists, distant objects do not look the same as objects that are close-up. It is proposed that an algorithm can be designed to make the ink intensity and blur level be affected by the depth map.

The artificial intelligence of moving objects can be strengthened. The result of this research project only applies basic AI to moving objects. In game applications, there can be more interaction between players and moving objects. Also, other game elements can be added to make the result more enjoyable. Finally, consideration should be given to the idea that the rendering pipeline can even be applied in some commercial game engines that would use the Chinese ink painting style in a video game.

There are many parallels between the algorithm described in this thesis and an artificial neural network. Both the internal coloring and silhouette processing involves computing linear combinations of pixel values in a neighborhood (specifically, a convolution), and then passing the results through a nonlinear activation function. This suggests the possibility of learning ink painting styles by example. Such a technique would likely require a large amount of training data. The feasibility of this should be explored in future work.

LIST OF REFERENCES

LIST OF REFERENCES

- Appel, A. (1967). The notion of quantitative invisibility and the machine rendering of solids. *Proceedings of the A.C.M. National Meeting*, 387–393.
<http://doi.org/10.1145/800196.806007>
- Baxter, W., Wendt, J., & Lin, M. C. (2004). IMPaSTo: A Realistic, Interactive Model for Paint. *Proceedings of the Third International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2004)*, 1(212), 45–56.
<http://doi.org/http://doi.acm.org/10.1145/987657.987665>
- Chen, Z., Kim, B., & Wang, H. (2015). Wetbrush : GPU-based 3D Painting Simulation at the Bristle Level, 34(6), 1–11.
- Chu, N. S. H., & Tai, C. L. (2004). Real-time painting with an expressive virtual Chinese brush. *IEEE Computer Graphics and Applications*, 24(5), 76–85.
<http://doi.org/10.1109/MCG.2004.37>
- Chu, N. S. H., & Tai, C. L. (2005). MoXi: Real-Time Ink Dispersion in Absorbent Paper. *ACM Transactions on Graphics, Proceedings of ACM SIGGRAPH 2005 (Los Angeles, CA, July 31--August 4, 2005)*, 24(3), 504–511.
<http://doi.org/http://doi.acm.org/10.1145/1073204.1073221>
- Danner, S. M., & Winklhofer, C. J. (2008). Cartoon Style Rendering. *Computer*, 14. Retrieved from
<https://www.cg.tuwien.ac.at/courses/Seminar/WS2007/comicstyle.pdf>
- Deng, L., Hinton, G., & Kingsbury, B. (2013). New types of deep neural network learning for speech recognition and related applications: an overview. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8599–8603.
<http://doi.org/10.1109/ICASSP.2013.6639344>
- Dong, L., Lu, S., & Jin, X. (2014). Real-time image-based chinese ink painting rendering. *Multimedia Tools and Applications*, 69(3), 605–620. <http://doi.org/10.1007/s11042-012-1126-9>
- Dooley, D., & Cohen, M. F. (1990). Automatic illustration of 3D geometric models: surfaces. *Proceedings of the First IEEE Conference on Visualization: Visualization '90*, 77–83. <http://doi.org/10.1109/VISUAL.1990.146395>

- Gooch, A., Gooch, B., Shirley, P., & Cohen, E. (1998). A non-photorealistic lighting model for automatic technical illustration. *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, 447–452. <http://doi.org/http://doi.acm.org/10.1145/280814.280950>
- Gregor, K., Danihelka, I., Graves, A., Jimenez Rezende, D., & Wierstra, D. (2015). DRAW: A Recurrent Neural Network For Image Generation. *Icml-2015*, 1462–1471.
- Hertzmann, A. (1999). Introduction to 3D Non-Photorealistic Rendering : Silhouettes and Outlines. *Non-Photorealistic Rendering. SIGGRAPH*, 99, 1–14. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.9731&rep=rep1&type=pdf>
- Huang, S. W., Way, D. L., & Shih, Z. C. (2003). Physical-Based Model of Ink Diffusion in Chinese Ink Paintings. *Proceedings of WSCG 2003*. Retrieved from http://wscg.zcu.cz/wscg2003/Papers_2003/H89.htm
- Isenberg, T., Freudenberg, B., Halper, N., Schlechtweg, S., & Strothotte, T. (2003). A Developer's Guide to Silhouette Algorithms for. *IEEE Computer Graphics and Applications*, (August), 28–37.
- Isenberg, T., Neumann, P., Carpendale, S., Sousa, M. C., & Jorge, J. A. (2006). Non-photorealistic rendering in context. *Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering - NPAR '06*, (2005-805-36), 115. <http://doi.org/10.1145/1124728.1124747>
- Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, {ACL} 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, 655–665. Retrieved from <http://aclweb.org/anthology/P/P14/P14-1062.pdf>
- Kalnins, R. D., Markosian, L., Meier, B. J., Kowalski, M. a., Lee, J. C., Davidson, P. L., ... Finkelstein, A. (2002). WYSIWYG NPR: drawing strokes directly on 3D models. *ACM Transactions on Graphics*, 21(3), 755–762. <http://doi.org/10.1145/566654.566648>
- Markosian, L., Kowalski, M. A., Goldstein, D., Trychin, S. J., Hughes, J. F., & Bourdev, L. D. (1997). Real-time nonphotorealistic rendering. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques SIGGRAPH 97* (Vol. 31, pp. 415–420). ACM Press/Addison-Wesley Publishing Co. <http://doi.org/10.1145/258734.258894>
- Mitani, J., Suzuki, H., & Kimura, F. (2002). From Geometric Modeling to Shape Modeling. *From Geometric Modeling to Shape Modeling*, 80(IFIP — The International Federation for Information Processing), 85–98. <http://doi.org/10.1007/978-0-387-35495-8>

- Nahar, K. (2012). Artificial Neural Network. *COMPUSOFT : International Journal of Advanced Computer Technology*, 1(2), 25–27.
- Ngene, B. U., Agunwamba, J. C., Nwachukwu, B. A., & Okoro, B. C. (2015). The Challenges to Nigerian Rain Gauge Network Improvement Department of Civil Engineering , University of Nigeria , Nsukka , Enugu State , Department of Civil Engineering , Federal University of Technology , Owerri , Imo State , Nigeria, 7(4), 68–74.
- Rössl, C., & Kobbelt, L. (2000). Line-art rendering of 3D-models. In *Proceedings - Pacific Conference on Computer Graphics and Applications* (Vol. 2000-Janua, pp. 87–96). <http://doi.org/10.1109/PCCGA.2000.883890>
- Sun, M. J., Sun, J. Z., & Yun, B. (2005). Physical modeling of “Xuan” paper in the simulation of Chinese Ink-Wash Drawing. *Proceedings of the Conference on Computer Graphics, Imaging and Vision: New Trends 2005*, 2005, 317–322. <http://doi.org/10.1109/CGIV.2005.59>
- Taormina, R., Chau, K., & Sethi, R. (2012). Artificial neural network simulation of hourly groundwater levels in a coastal aquifer system of the Venice lagoon. *Engineering Applications of Artificial Intelligence*, 25(8), 1670–1676. <http://doi.org/10.1016/j.engappai.2012.02.009>
- Wang, A. Y., Tang, M., & Dong, J. X. (2004). A Survey of Silhouette Detection Techniques for Non-Photorealistic Rendering. *Proceedings of the Third International Conference on Image and Graphics (ICIG'04, December 18--20, 2004, Hong Kong, China)*, 2, 434–437. <http://doi.org/http://doi.ieeecomputersociety.org/10.1109/ICIG.2004.28>
- Xu, T., Yang, L., & Wu, E. (2012). Stroke-based real-time ink wash painting style rendering for geometric models. *SIGGRAPH Asia 2012 Technical Briefs on - SA '12*, 1(212), 1–4. <http://doi.org/10.1145/2407746.2407765>
- Yang, L., Xu, T., & Li, X. (2011). An Image-based Approach in Animating Painting Procedure of Chinese Ink Painting. *Stroke*, 1(1), 1–4.
- Yuan, M., Yang, X., Xiao, S., & Ren, Z. (2007). GPU-Based Rendering and Animation for Chinese Painting Cartoon. In *Proceedings of Graphics Interface 2007 (GI 2007, May 28-30, 2007, Montr{é}al, Canada)* (Vol. 234, pp. 57–61). <http://doi.org/http://doi.acm.org/10.1145/1268517.1268529>

APPENDICES

Appendix A Figure 4.6 with high resolution

The silhouette of the mesh is more continuous than shown in images in the left and middle of Figure 4.6.



Figure A.1 Figure 4.6 with higher resolution

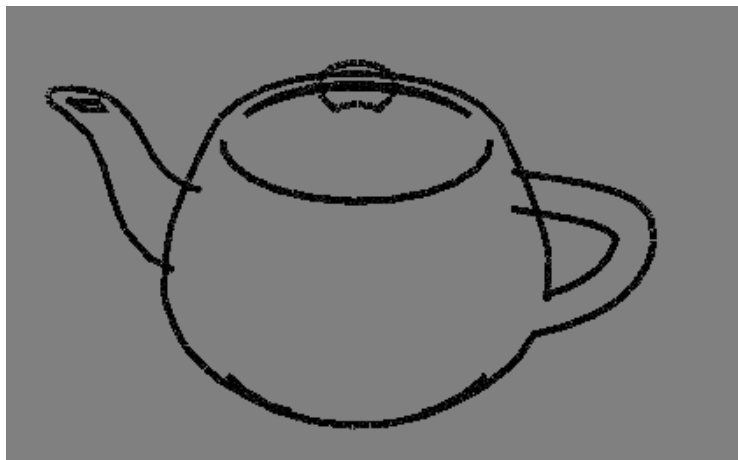
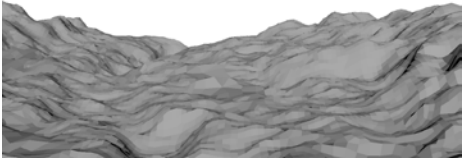
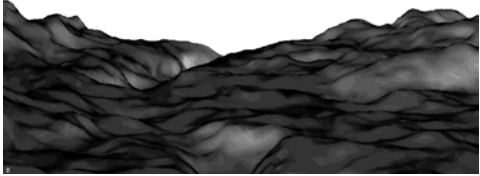
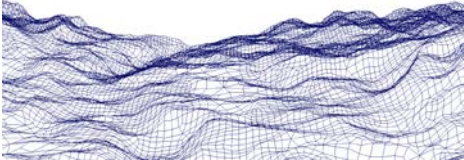
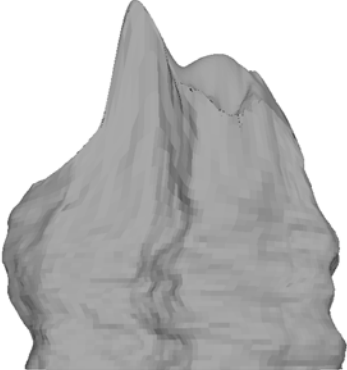
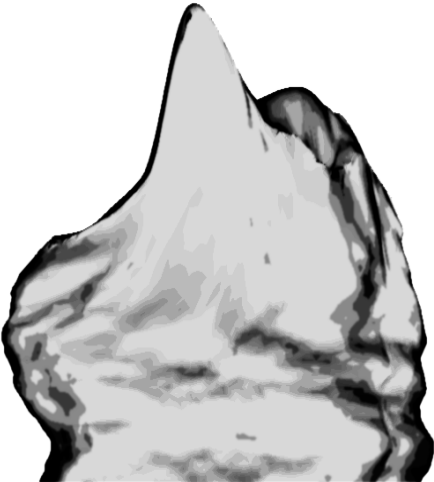
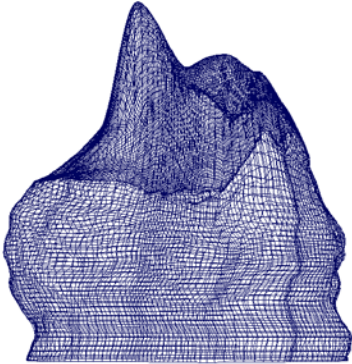
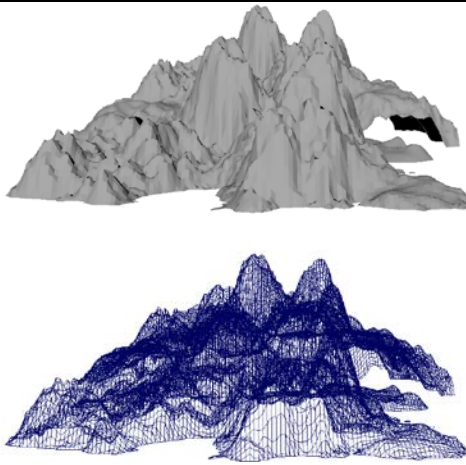
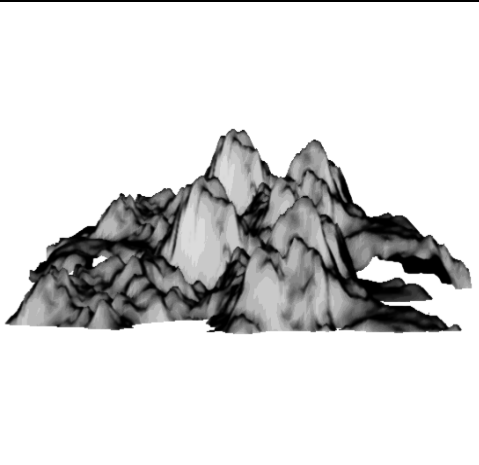




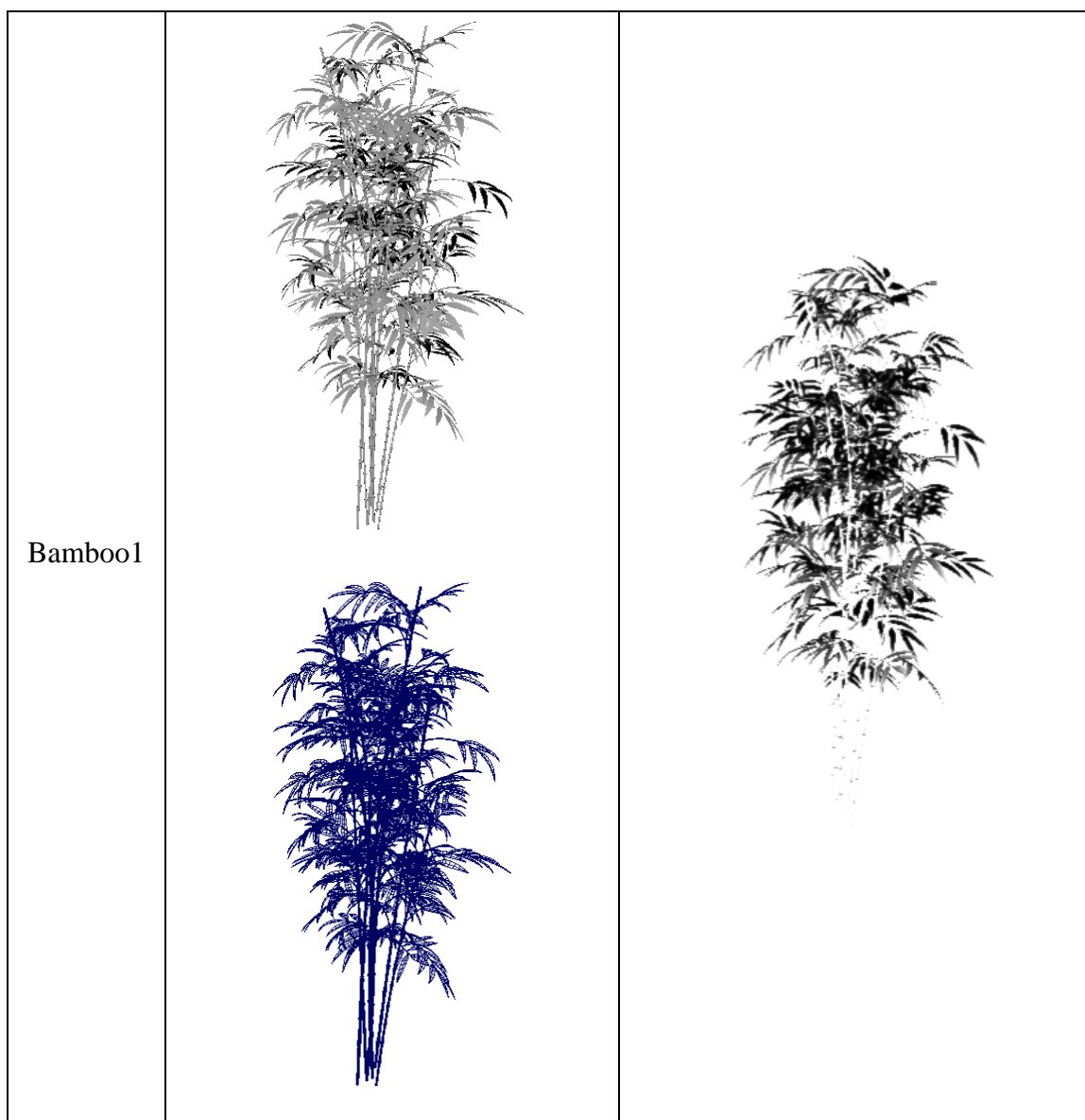
Figure A.2 Figure 4.6 with higher resolution

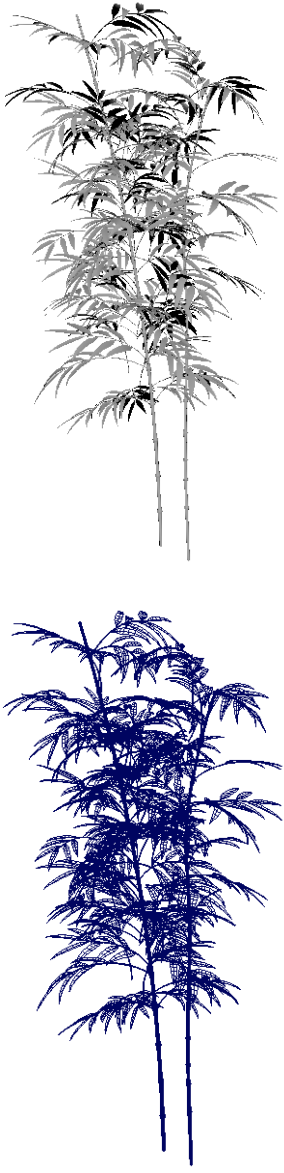

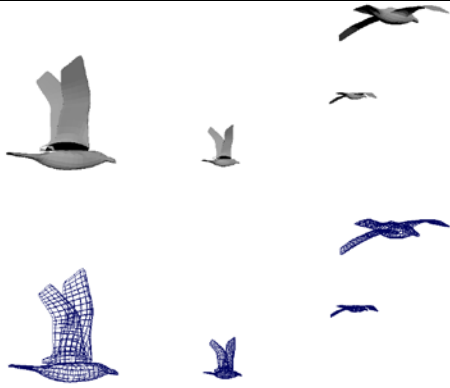

Appendix B Results of Rendering Single Meshes

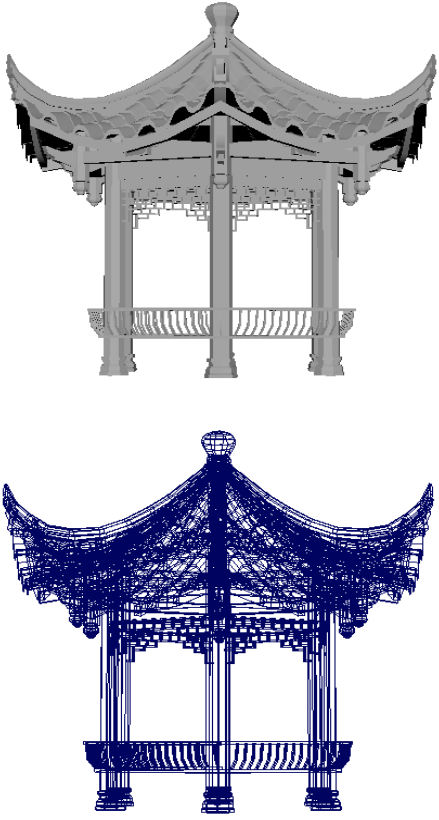
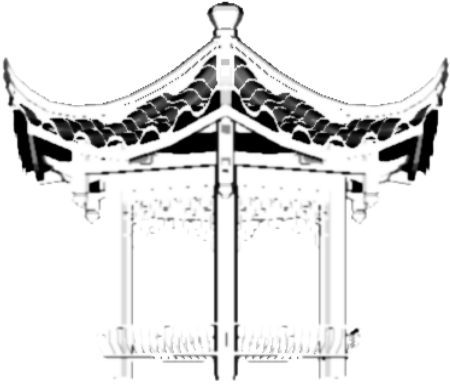
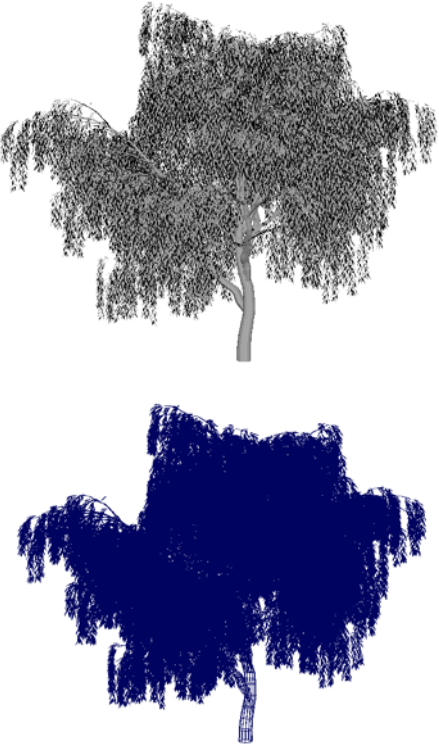

Table 2 Rendering single meshes

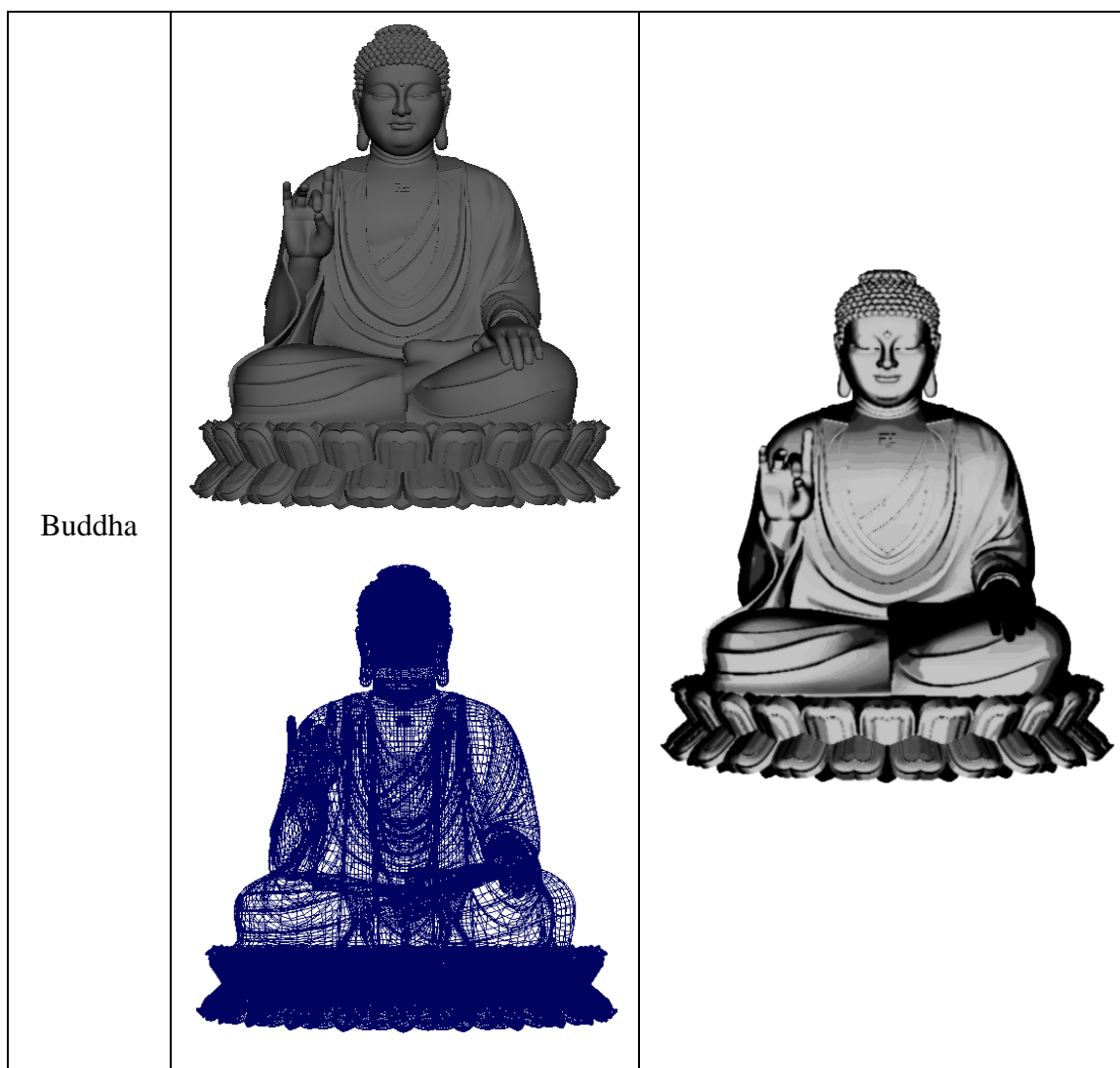
Mesh name	Original mesh	Rendering
Terrain		
		
Mountain		
		

Mountains	 Two 3D mountain models are shown. The top model is a solid, light gray mountain range with several peaks and a small cave-like opening on the right. The bottom model is a blue wireframe version of the same mountain range, showing the underlying mesh structure.	 A 3D mountain model with a grayscale gradient. The peaks are white, and the slopes and valleys transition through shades of gray to black, creating a sense of depth and shadow.
Bamboo	 Two 3D bamboo models are shown. The top model is a light gray bamboo plant with many thin, upright stalks and clusters of leaves. The bottom model is a blue wireframe version of the same bamboo plant, showing the underlying mesh structure.	 A 3D bamboo model with a grayscale gradient. The stalks and leaves are rendered with varying shades of gray, giving it a more realistic appearance than the solid models.



Bamboo2	 <p>This panel contains two illustrations of bamboo plants. The top illustration is a black and white line drawing of a bamboo stalk with several branches and leaves. The bottom illustration is a blue wireframe version of the same bamboo plant, showing the structural framework of the stalk and leaves.</p>	 <p>This panel contains a single black and white illustration of a bamboo plant, similar in style to the one in the middle panel, showing a stalk with multiple branches and leaves.</p>
Birds	 <p>This panel displays a collection of birds in flight. It includes a large white bird with its wings spread, a smaller white bird, and several smaller blue birds. The birds are arranged in a scattered pattern, suggesting a flock in flight.</p>	 <p>This panel displays a collection of birds in flight, similar to the middle panel. It features a large white bird, a smaller white bird, and several smaller blue birds, all shown in various flight poses.</p>

<p>Pavilion</p>		
<p>Willow</p>		



Appendix C Gaussian Blur

The Gaussian blur equations and smooth step functions are coded in the fragment shader.

The calculation done by the following codes is for the 1D Gaussian blur equation:

$$G(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}.$$

This equation is used to compute a weight for each pixel in the

Gaussian kernel.

```
float GaussianLoad1D(float x) {
    float ret = (1.0 / (2.0*3.1415926)) * pow(2.71828, -(x*x / 2.0));
    return ret;
}
```

The calculation done by the following codes is for the 2D Gaussian blur equation:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

```
float GaussianLoad2D(int x, int y, float delta2){
    float distance = float(x)*float(x) + float(y)*float(y);
    float ret = (1.0 / (2.0 * delta2 * 3.1415926)) * pow(2.71828, -(distance /
(2.0 * delta2)));
    return ret;
}
```

The following code shows the process of calculating the value of “diffuse” with 1D Gaussian blur applied for interior shading. The variable “diffuse” is computed based on the light vector and the normal vector.

```
float InteriorGaussianBlur(float ndotl){
    float ret = 0.0, temp = 0.0;
    for(float i=-0.3; i<=0.3; i=i+0.1){
        temp = ndotl + i;
        if(temp <= 0.25){
            ret += GaussianLoad1D(i) * 1.0;
        }else if(temp <= 0.55){
            ret += GaussianLoad1D(i) * 0.7;
        }else if(temp <= 0.8){
            ret += GaussianLoad1D(i) * 0.4;
        }else{
            ret += GaussianLoad1D(i) * 0.1;
        }
    }
    return ret;
}
```

```
}
```

The following code shows the process of calculating the value of “edge” with 1D Gaussian blur applied for silhouette shading. The variable “edge” is computed based on the view vector and the normal vector.

```
float SilhouetteGaussianBlur(float ndotv){
    float ret = 0.0, temp = 0.0;
    //float threshold = 0.5;

    for(float i=-0.3; i<=0.3; i=i+0.1){
        float temp = min(ndotv + i, 1.0);
        if(temp > threshold){
            ret += GaussianLoad1D(i) * 1.0;
        }
    }
    return ret;
}
```

The following code shows the process of applying the 2D Gaussian blur function to the silhouette texture. The level of blur depends on the parameter “kernelWidth” which refers to the Gaussian kernel. When the value of “kernelWidth” equals 7, the final scene of this research project provided the best performance.

```
vec4 GaussianBlur(int kernelWidth, sampler2D tex){
    vec4 ret = vec4(0.0, 0.0, 0.0, 1.0);
    if(kernelWidth == 1){
        ret = texture(tex, texCoord);
        return ret;
    }
    float delta2 = (float(kernelWidth)-1.0) / 2.0 / 3.0;
    delta2 = delta2 * delta2;
    for(int i=-kernelWidth/2; i<=kernelWidth/2; ++i){
        for(int j=-kernelWidth/2; j<=kernelWidth/2; ++j){
            vec4 col = texelFetch(tex, ivec2(texCoord.s * 1200.0,
texCoord.t * 1000.0) + ivec2(i, j), 0); // texelFetch(tex, ivec2(gl_FragCoord),
0);
            ret += col * GaussianLoad2D(i, j, delta2);
        }
    }
    ret.a = 1.0;

    return ret;
}
```

Appendix D Multi-pass Rendering

The following code is in the display function which is a callback function from OpenGL Utility Toolkit (GLUT).

```
// pass 1: render the interior part to a texture
RenderToTexture(inkPaintingShadingProgram, interiorTexID, interiorFBO, P,
eyeView, 0);
// pass 2: render the silhouette to a texture
RenderToTexture(inkPaintingShadingProgram, silhouetteTexID, silhouetteFBO, P,
eyeView, 1);
```

The scene is rendered to the interior and silhouette texture independently in pass 1 and pass 2, and then integrated in the third pass. The following code is in the fragment shader for rendering the third pass. 2D Gaussian blur and the smooth step function are applied in pass 3.

```
subroutine void RenderPassType();
subroutine uniform RenderPassType RenderPass;

subroutine (RenderPassType)
void InteriorRender() {
    fragColor = texture(interiorTex, texCoord);
}
subroutine(RenderPassType)
void SilhouetteRender() {
    fragColor = GaussianBlur(kernelWidth, silhouetteTex);
}
subroutine(RenderPassType)
void BothRender() {
    vec4 siColor = GaussianBlur(kernelWidth, silhouetteTex);
    siColor.r = smoothstep(0.1, 0.8, siColor.r);
    siColor = vec4(vec3(siColor.r), 1.0);
    vec4 inColor = texture(interiorTex, texCoord);
    fragColor = vec4(siColor.r * inColor.r, siColor.g * inColor.g, siColor.b *
inColor.b, 1.0);
    // smooth step
    fragColor.r = smoothstep(edge0, edge1, fragColor.r);
    fragColor = vec4(vec3(fragColor.r), 1.0);
}

void main(void) {
    RenderPass();
}
```

Appendix E Mesh Information

The data structure for static meshes is shown below:

The data types beginning with “ai” are used by the Open Asset Import Library (Assimp) for loading meshes. Each mesh has its own vertex array object (VAO) and vertex buffer objects (VBO) to store its attributes including position, normal, and texture coordinate. Each vertex has an index which is stored in an index buffer.

```
struct MeshData {
    unsigned int mVao;
    unsigned int mVboVerts;
    unsigned int mVboNormals;
    unsigned int mVboTexCoords;
    unsigned int mIndexBuffer;
    float mScaleFactor;
    unsigned int mNumIndices;
    const aiScene* mScene;
    aiVector3D mBbMin, mBbMax;
};
```

For meshes with skeleton animations, each bone has an ID. Each vertex can be affected by at most 16 bones with correlated weights stored in VertexBoneData. The IDs of the involved bones are also stored in VertexBoneData as shown below:

```
#define NUM_BONES_PER_VERTEX 16
struct BoneInfo {
    aiMatrix4x4 BoneOffset;
    aiMatrix4x4 FinalTransformation;
    BoneInfo() {
        aiMatrix4x4::Scaling(aiVector3D(0.0f), BoneOffset);
        aiMatrix4x4::Scaling(aiVector3D(0.0f), FinalTransformation);
    }
};
struct VertexBoneData {
    unsigned int IDs[NUM_BONES_PER_VERTEX];
    float Weights[NUM_BONES_PER_VERTEX];
    VertexBoneData() {
        memset(IDs, 0, sizeof(IDs));
        memset(Weights, 0, sizeof(Weights));
    };
    void AddBoneData(unsigned int BoneID, float Weight);
};
```